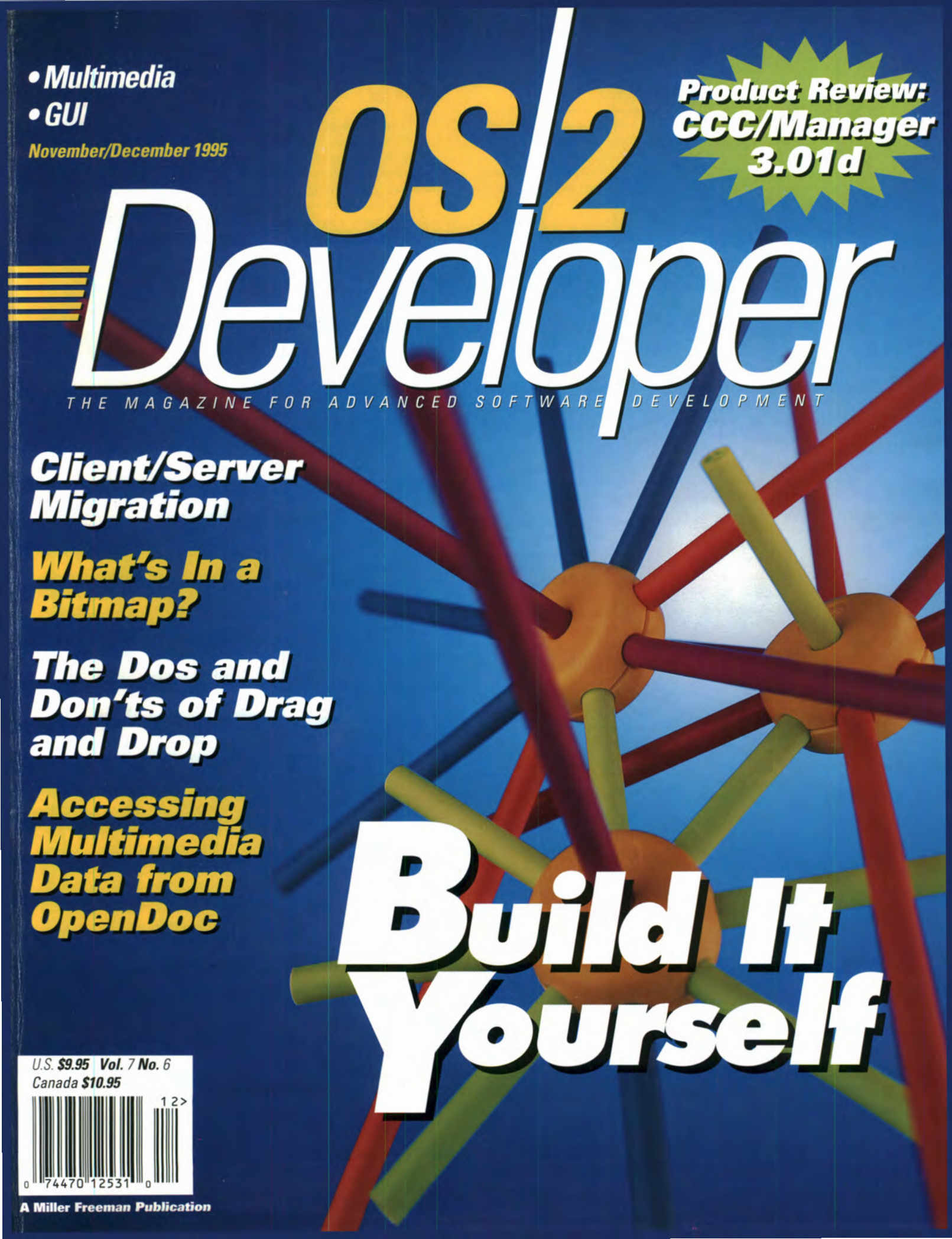

• Multimedia
• GUI
November/December 1995
os/2 Developer
THE MAGAZINE FOR ADVANCED SOFTWARE DEVELOPMENT
Product Review: CCC/Manager 3.01d
Client/Server Migration
What's In a Bitmap?
The Dos and Don'ts of Drag and Drop
Accessing Multimedia Data from OpenDoc
Build It Yourself
U.S. $9.95 Vol. 7 No. 6
Canada $10.95
12>
0 74470 12531 0
A Miller Freeman Publication

The verdict is: One's a pleasure, two are useful, and one's a pain.

INFOWORLD *Scores run from 7.2 to 2.7, from worthy to worthless.*

	zApp Developer's Suite	C++View 1.0.1	Zinc Application Framework 4.0	Windo 2.0a
✓ Installation & Configuration (75)	46.87	46.87	37.50	37.50
✓ Generating a Skeleton App. (125)	125.00	31.25	62.50	31.25
✓ Interface Design (150)	112.50	75.00	93.75	0.00
✓ Event Processing (125)	78.12	78.12	62.50	62.50
✓ Enhancement & Maintenance (125)	93.75	78.12	78.12	31.25
✓ Portability (150)	75.00	75.00	112.50	37.50
✓ Documentation (100)	62.50	25.00	62.50	0.00
✓ Support (100)	100.00	100.00	75.00	50.00
✓ Pricing (50)	31.25	31.25	50.00	25.00
FINAL SCORE	7.2	5.4	6.3	2.7

Dr. Dobb's JOURNAL *There's a saying that good design is invisible. As Ward Cunningham puts it, "Good class libraries whisper the design in your ear." This is the case with zApp, which straightforwardly provides all the usual classes you'd expect in building event-drawn GUI programs: a main application class, a small hierarchy of event handling classes, classes for graphic display, window-containing classes, and the usual GUI widgets (push buttons, check box, list box, and so on). There are no radical concepts or unpleasant surprises, greatly easing any learning curve associated with a brand-new API.*

Ray Valdés - October, 1992

ProgramNOW *The documentation provided with zApp is little short of superb. zApp 2.0 is a professional product which turns platform-independence into a practical proposition by its efficient, versatile and function-rich implementation.*

EXE *When presenting zApp with their Editor's Choice Award, Willie Watts explained why he "plumped for zApp": Of all the C++ encapsulations of Windows in all the bars in the world, we both felt this was the best attempt we had seen, and was something of a landmark in the use of this important language.*

JOURNAL OF OBJECT-ORIENTED programming *The class design is the most important and most subjective aspect of a class library, and I think zApp's is superb. zApp encapsulates the entire underlying native API. In other words, if you want a Windows dialog, you get a Windows dialogue, not something that looks like one.*

Windows Tech JOURNAL *The 1992 Star Tech Award for C/C++, and Application Frameworks goes to Inmark Development Corp's zApp V.2.0. zApp is simply the best designed application framework on the market. Its well-conceived classes make Windows programming an easier and more manageable process-and even fun. If C++ is your language of choice, then you should be using zApp. Period.*

COMPUTER SHOPPER *For general-purpose applications that aren't limited to text, basic graphics, and dialog boxes, however, I suggest evaluating zApp first. Features like advanced graphics, printing support, DDE, compatibility with third-party resource tools, and the ability to incorporate custom controls can be hard to live without...*

Reprinted from "COMPUTER SHOPPER" Sept. 1993
Copyright© 1993 Ziff-Davis Publishing Company

PC MAGAZINE *As long as you write your applications using zApp objects, you can create programs for one compiler or environment and recompile them with another compiler for a different environment without any changes... zApp's object hierarchy remains the most comprehensive I've yet seen in an application framework.*

Reprinted from "PC MAGAZINE" Dec. 22, 1992
Copyright© 1992 Ziff-Davis Publishing Company

C++REPORT *Marketing people say things like "zApp is great. It contains over 200 classes." That looks good in a slick and glossy brochure, but it leaves the fundamental question unanswered: Are they good classes? They are. zApp's classes are well-implemented and easy to use. The library covers the full breadth of contemporary functionality and offers portability across most major operating systems.*

PC TECHNIQUES *What I've noticed about the company behind the product is that Inmark's customer technical support is outstanding (usually a returned call within four hours), their BBS is full of great information, and their attention to quality is stellar. I wish all software tool providers did as well.*

PCWEEK *Mark Brittingham, a user-interface specialist with AT&T Bell Labs in Middletown, NJ, chose zApp over other frameworks. "You can see it's built by someone who builds applications,"*

Reprinted from "PC WEEK" July 19, 1993
Copyright© 1993 Ziff-Davis Publishing Company

SOFTWARE Development *With the addition of a visual screen builder, the final barrier to zApp productivity has fallen. For quick, form-based applications, zApp Factory makes C++ productive enough to be competitive with the interpreted languages. For more serious applications, the zApp framework is a superior application framework. I would use it even if I were just programming for Windows, but the fact that it compiles for Windows, Win/NT, OS/2, UNIX, and X/Motif, is a nice little bonus.*

zApp

Award winning developers use Award winning tools.

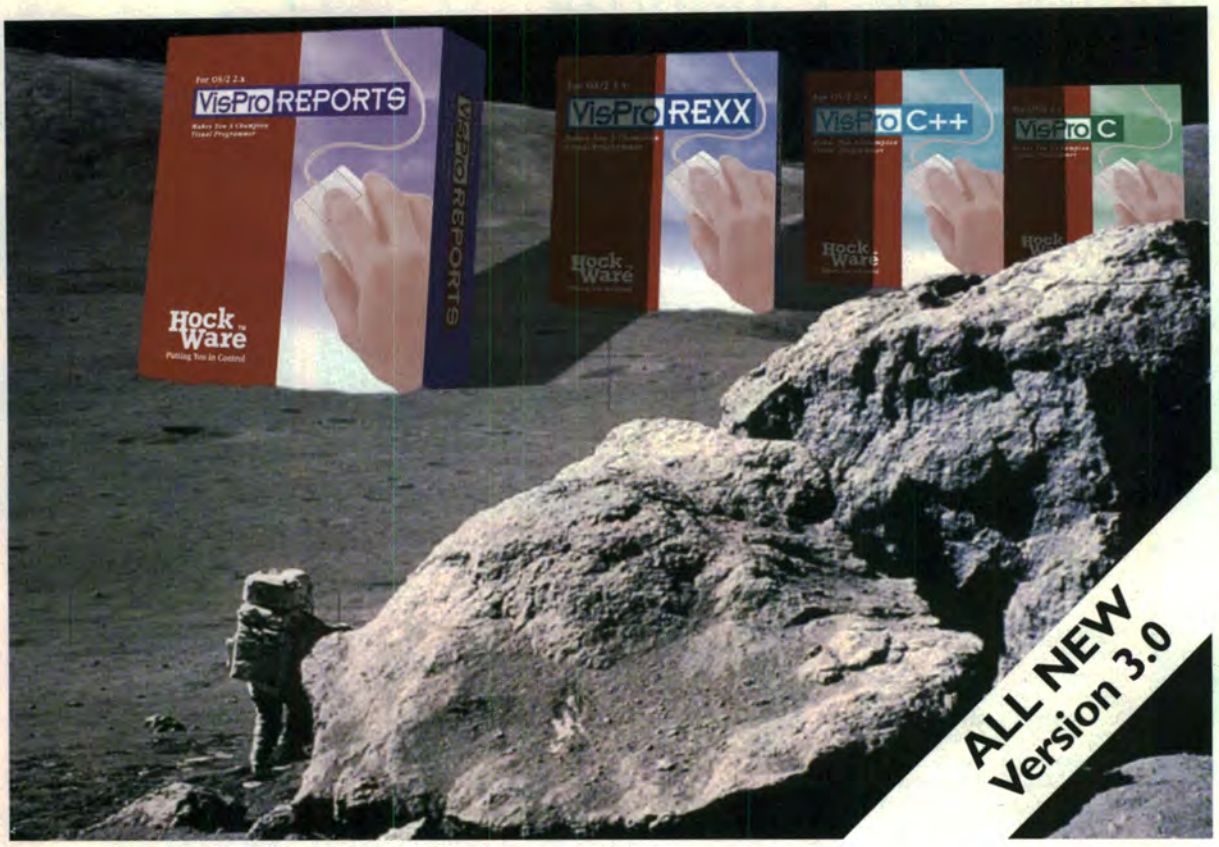
With the award winning zApp® Developer's Suite, you can quickly build state-of-the-art C++ applications that run across fourteen of today's most popular operating systems. The zApp Developer's Suite contains the latest in drag-and-drop design, prototyping and testing, code generation, and powerful GUI objects. Best of all this powerful technology sits on top of zApp, the industry leading application framework in use by tens of thousands of developers worldwide. **For a free demo call: 1-800-346-6275**

INFOWORLD TOP SCORE

INMARK

Inmark Development Corporation 2065 Landings Drive, Mountain View, CA 94043 (800) 346-6275, (415) 691-9000 Fax: (415) 691-9099
In the U.K. and Scandinavia, call **PTS/Software Plus** at +44 (0) 928 579900, In France, call **PTS/Software Plus** at (05) 908194, In Germany, call **ESM Software** at 07702-9256-0, In Italy, call **Silicon Valley On-Line** at (049) 8719820, In Australia, call **Micro Way** at (03) 580-1333, **BBS:** 415-691-9990 • **Internet:** info@inmark.com • **CompuServe:** GO INMARK

It Took Teamwork to Reach the Moon



With the all new VisPro 3.0, your OS/2 programming team can work together, *at a price that's down-to-earth*

For years, the VisPro family of visual programming products has made OS/2 programming easy with Workplace Shell-enabled drag and drop programming. Our responsive and FREE technical support never leaves customers drifting in space. Now, whether you're part of a large OS/2 development crew or flying solo, the all-new VisPro/REXX Gold, VisPro/C and VisPro/C++ will guarantee you a smooth client/server programming ride.

Team Development
Take a seat at mission control with VisPro 3.0's Team Administrator. Compare change levels and project versions with ease, rollback changes, and monitor development progress. Team development also gives you automatic change logging, form shadowing, form locking and read-only browsing; all seamlessly integrated, so you don't need to be a rocket scientist.

Support for More Databases
With VisPro's entity/relationship Database Designer, you can now design and reverse-engineer IBM DB2, Watcom SQL, or any ODBC-enabled database. Generating DDL is fast and easy. VisPro/C and VisPro/C++ now generate native embedded SQL for Sybase, Oracle, IBM DB2 and Watcom SQL. VisPro 3.0 makes database programming easier than ever. With a simple drag and drop, you can instantly generate GUI panels for adding, changing, deleting and searching for rows in a table. The all new join and drill-down support makes creating complex queries a breeze.

Other Essential Gear
The VisPro/Paint image editor is included free to allow you to edit BMP, TIF, GIF, and many other image formats. The VisPro IPF Help File editor makes it easier to create online help for your applications.

Both VisPro/C and VisPro/C++ now support the IBM VisualAge C++ and MetaWare High C compilers. In addition, VisPro/C also supports the Borland and Watcom OS/2 compilers. VisPro/REXX includes hundreds of new APIs, Dynamic Data Exchange (DDE) support and an improved multi-threaded REXX debugger with support for conditional break points and animation.

The Complete VisPro Family

VisPro/REXX Gold 3.0	**$299**
VisPro/REXX Bronze	**$ 59**
VisPro/REXX Data Entry Object Pack	**$ 89**
VisPro/C or VisPro/C++ 3.0	**$299**
VisPro/Reports	**$199**
VisPro Development Suite 3.0	**$499**

Whether you've already discovered the VisPro family of visual programming products of you're just getting to know us, VisPro 3.0 will take you where you want to go.

Special Offer
Buy VisPro/REXX Gold and get the VisPro Image Processing Object Pack absolutely free.

HockWare™
Putting You in Control

HockWare Incorporated
P.O. Box 336
Cary. NC 27512-0336
919-380-0616
919-380-0757 FAX
Go HockWare on CompuServe
hockware@vnet.net on Internet
http://www.hockware.com/hockware

HockWare, VisPro/C, VisPro/C++, VisPro/REXX and VisPro/Reports are trademarks of HockWare Incorporated. All other company, product and brand names are trademarks and/or registered trademarks of their respective holders and are mentioned for reference purposes only. ©1995 HockWare Incorporated. All rights reserved.

Circle Reader Service Number 1

BEST SELLERS FROM THE Programmer's Paradise® CATALOG

Visual SlickEdit for OS/2— the intelligent programmer's editor

by MicroEdge, Inc.

This high-powered programmer's editor is completely customizable with the speed to ignite your productivity. Save time using BRIEF, Emacs, or VI emulations. Take off with Visual SlickEdit's GUI interface, built-in dialog editor, and C-style macro language. Features include: SmartPaste™; compiler error processing; syntax color-coding; expansion and indenting; and an on-line manual. 30 day risk-free trial!

Discount Price: **$219*** ** Offer ends December 31, 1995.*
Paradise No. MEVSE3100-EI

KopyKat Remote Control

by Hilgraeve, Inc.

Anything you can run on a PC with OS/2 you can now do remotely via LAN, Modems, or Serial Cable. You can run: the remote computer's entire OS/2 desktop; presentation manager programs—both 16- and 32-bit; command sessions or text mode...windowed or full screen; Microsoft Windows "seamless" or full screen; and DOS sessions or programs, windowed or full screen. Runs on Novell & IBM LANs.

Discount Price: **$119**
Paradise No. HGKKO3100-EI

For more details on the products featured on these pages call our 800 Number and we will fax you additional information.

HyperACCESS for OS/2

by Hilgraeve, Inc.

HyperACCESS for OS/2 is the complete, retail version of HyperACCESS Lite, the easy-to-use terminal program bundled with OS/2 Warp. HyperACCESS for OS/2 comes packed with exciting new graphical powers, a resource library listing thousands of ready-to-call online systems, and instant-benefit utilities that let even novice users reap immediate rewards from going online.

Discount Price: **$89**
Paradise No. HGHAS3100-EI

PRODUCT OF THE MONTH

IBM® VisualAge™ C++ Version 3.0

by IBM

IBM VisualAge C++ V3.0 combines the productivity-boosting visual programming capability of IBM VisualAge with robust, professional development tools from the popular IBM C Set++™ product. This powerful combo lets you visually build parts then assemble them into mission-critical, object-oriented programs.

CD & On-Line Docs — Discount Price: **$399**
Paradise No. IB30H1666-EI
Competitive Upgrade — Discount Price: **$289**
Paradise No. IB30H1776-EI

MORE HOT SELLERS!

	Paradise No.	Discount Price
Borland C++ 2.0 for OS/2	BOCOT8888-EI	$339
PREDITOR/2	CWPD23100-EI	$149
GammaTech Utilities	STGT03100-EI	$99
IBM Smalltalk 2.0	IBM14H0270-EI	$659
IBM VisualAge 2.0	IBM17H7495-EI	$1549
Mesa/2™	ADS2O3100-EI	$189
MKS Source Integrity for OS/2	MKSIO3100-EI	$360
OnCmd Xbase for OS/2	OLDCX3100-EI	$240
Opt Tech Sort	OPOTS0100-EI	$240
Prominare Designer	PRDGO3100-EI	$640
Watcom C/C++ V10.5 Competitive Upg.	WACCT8888-EI	$189

c-tree Plus®

by FairCom Corporation

DOS • WINDOWS • NT • UNIX • OS/2 • SUN • RS6000 • HP9000 • MAC • QNX • BANYAN • SCO. This well known, highly portable data management package has become established as the tool of choice for commercial development. Offering unprecedented data control, choose from direct low level access, ISAM level, or SQL access with the FairCom Server. Single-User, Multi-User, or optional Client/Server, ANSI Standard. Full Source. No Royalties.

FAIRCOM® since 1979

Discount Price: **$769**
Paradise No. FACT+3100-EI

DataTable for OS/2

by ProtoView Development Corp.

DataTable is a sophisticated spreadsheet control. Through a robust message-based API and numerous notification codes, developers can control every aspect of their application. Features include: virtual memory management; column sorting; column locking; check boxes and combo boxes; and colors and fonts may be set on a cell, row, column or table basis. New for OS/2, vertical window splitting.

Discount Price: **$476**
Paradise No. PDDTO3100-EI

Watcom VX•REXX Client/Server 2.1

by Watcom International

A visual development environment for OS/2. Powerful connection, query and chart objects allow you to access several databases, manipulate data and chart the results quickly and easily. Features include: drag-and-drop programming; bound controls; and professional multi-threaded, multi-windowed and drag-and-drop enabled application development.

Discount Price: **$281**
Paradise No. WATVC3100-EI

SPF/PC Version 4.0

by Command Technology Corp.

Provides a familiar environment on the PC for ISPF/PDF mainframe programmers. Includes: Modifiable Panels & Table Services; UNDO/REDO; Program Source Colorization; SUPERC file comparison; vertical splits; mouse support; 64,000 max record length; 132-column support; and 20+ new primary commands. SPF/PC features: full 3270 compatibility; ISREDIT macros via REXX; ASCII/EBCDIC file editing; COBOL workbench integration; CUT/PASTE; fully mappable keyboard; and keyboard macros.

Discount Price: **$199**
Paradise No. CTSP43100-EI

Phone order
800 445 7899
FAX order
908 389 9227
International Sales
908 389 9229

Programmer's Paradise®
1163 Shrewsbury Avenue
Shrewsbury, NJ 07702-4321
Call for shipping charges/return policies.
PRICES SUBJECT TO CHANGE.

Order Now! 800-445-7899

Circle Reader Service Number 2

OS/2 Developer
NOVEMBER/DECEMBER 1995
THE MAGAZINE FOR ADVANCED SOFTWARE DEVELOPMENT

EDITOR'S COMMENTS
Serving Our Clients **5**

OS/2 TOOLBOX
CCC/Manager for OS/2 **6**

GUI CORNER
What's In a Bitmap? **12**

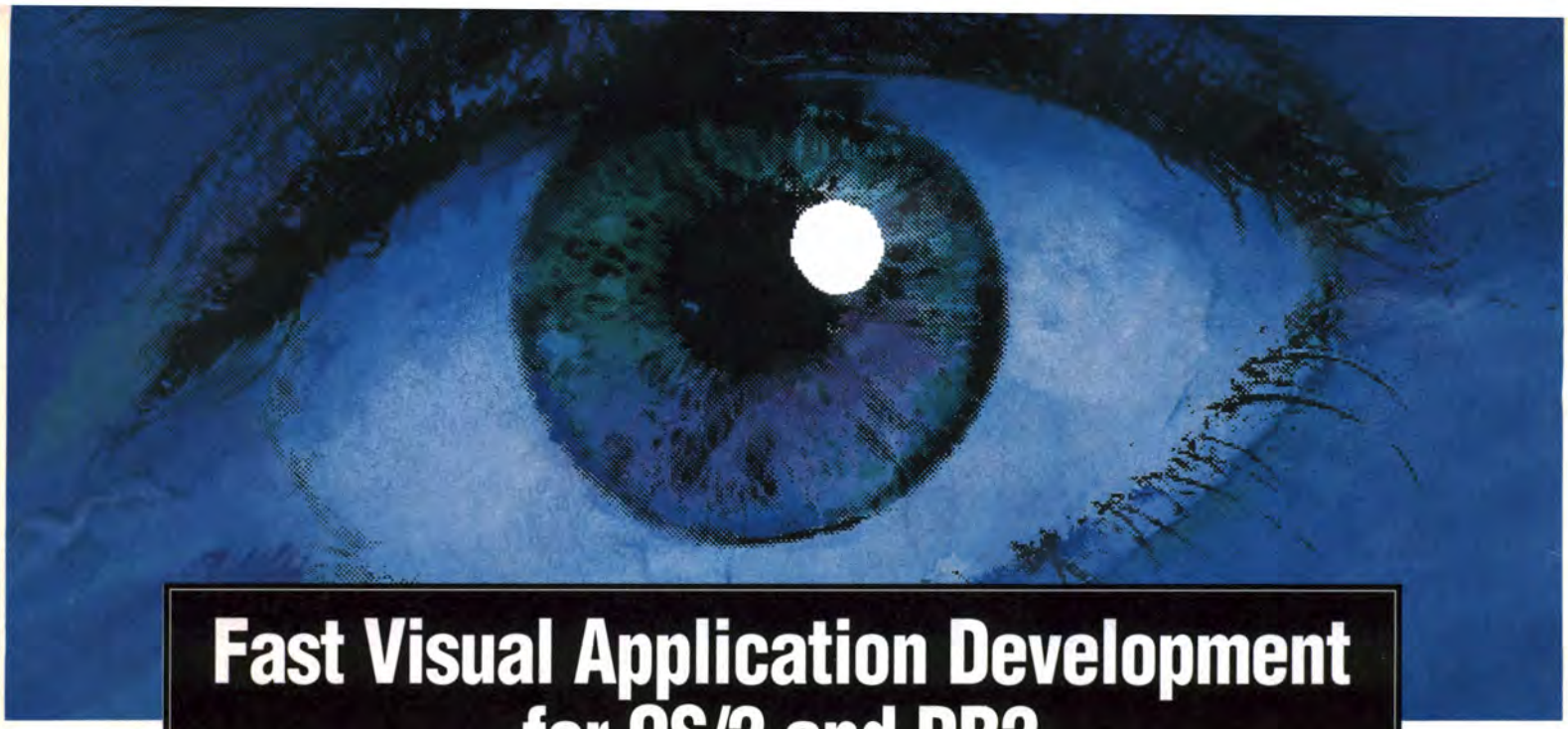
PROGRAMMING INSIDER
Drag Me, Drop Me, Treat Me Like an Object **20**

CLIENT/SERVER DEVELOPMENT
Client/Server Migration—The Office Client Rollout **24**
By Steve Krantz
OS/2 Server Design Using Named Pipes **30**
By Michael Barillier
When CORBA Objects Meet Transactions **35**
By Robert Orfali, Dan Harkey, and Jeri Edwards
DB2/2 Communication Using Novell's NETBIOS Over IPX **40**
By Jeffrey Altman
Client/Server Tools Buyers Guide **64**

GUI
A Multicolumn Drag/Drop Listbox **46**
By Mark McMillan

MULTIMEDIA
Accessing Multimedia Data from OpenDoc **58**
By Scott Broussard

PRODUCT WATCH
New Tools for OS/2 **70**



Fast Visual Application Development for OS/2 and DB2

If you're looking for fast and easy application development for OS/2, then take a look at the award-winning Watcom VX•REXX visual development environment. VX•REXX lets you build applications to exploit the graphical user interface, multi-threading, and multi-processing power of OS/2. VX•REXX Client/Server Edition gives you the added power to access DB2 or other database systems, manipulate the data, and chart the results at lightning speed.

"We like VX•REXX. Using it for development feels like driving a Porsche: it's fast, it's compact, everything's in the right place, and it makes us look good, too." Peter Coffee, PC WEEK

Designed to Meet Your Needs.

Watcom VX•REXX combines a project management facility, visual designer and an interactive debugger to deliver a highly productive visual development environment. The Client/Server Edition includes additional powerful objects so you can rapidly create rich GUI database applications. You can create OS/2 client applications which connect to DB2/2 or DB2/6000. Use IBM's DRDA support on OS/2 to access DB2 for MVS, DB2/400 for AS/400, and DB2/VSE and VM (SQL/DS) for VM and VSE. Also supported are Watcom SQL and ODBC-enabled databases†.

"Overall, this edition of VX•REXX for OS/2 is an outstanding visual client/server development platform." Nicholas Petreley, InfoWorld

Point. Click. And Presto!

To create an application you draw user interface objects, customize their properties using standard OS/2 notebooks, and define their event code using powerful drag-and-drop programming. To add database access just draw a query object, visually design a SQL query, press OK and presto— your window is automatically populated with objects that are bound to your query to display, update and search your data.

"Drag-and-drop nirvana." Nicholas Petreley, InfoWorld

Give Your Data a Whole New Image.

Energize your applications by displaying your data in a 3D chart. The Client/Server Edition gives you more than a dozen chart types to choose from, along with over 150 display options. You also get complete support for run-time events so you can bring new drama to your data by making your chart interactive.

"VX•REXX is a must buy." Jacques Surveyer, ComputerWorld

Standard or Client/Server Edition— Which one is for you?

To start creating powerful OS/2 GUI applications right away, order your copy of Watcom **VX•REXX Standard Edition** for just....**$99***

Or, to start creating rich client/server database applications, order Watcom **VX•REXX Client/Server Edition** for just.....**$299***

- Over 2 dozen objects, including CUA'91 containers, notebooks, pop-up menus and more
- Integration and control of existing applications through DDE, keystokes or REXX API's
- Easy to learn event-driven programming model with complete on-line documentation
- Support for professional multi-threaded, multi-windowed and drag-and-drop enabled applications
- Code reusability through section and file sharing
- Graphically create CUA'91 Presentation Manager objects, quickly customize their properties, and easily attach REXX procedures
- Package your application as an EXE or PM macro for royalty-free distribution

Ready! for OS/2 · Available and Ready for IBM LAN · PC Magazine Editors' Choice February 8, 1994 Watcom VX-REXX, Version 1 · OS/2 Professional "The best new development tool of 1993" · Software Development Productivity Award 1993 · OS/2 Magazine Editors' Choice Award 1994 Watcom VX•REXX V2.0

1-800-265-4555

Watcom
A Powersoft Company

Watcom International 415 Phillip Street, Waterloo, Ontario, Canada N2L 3X2 Tel. (519) 886-3700 Fax (519) 747-4971 *Prices and specifications are subject to change without notice. Price does not include freight and taxes where applicable. Prices quoted in US dollars.
† ODBC drivers are available from INTERSOLV, Inc. Watcom, the Lightning Device, and VX•REXX are trademarks of Watcom International Corporation. Other trademarks are properties of their respective owners. ©Copyright 1994 Watcom International Corporation.

Circle Reader Service Number 3

OS/2 Developer
THE MAGAZINE FOR ADVANCED SOFTWARE DEVELOPMENT

EDITOR *Dick Conklin*
Compuserve: 76711,1005

MANAGING EDITOR *Theresa Mori*
tmori@mfi.com

EDITORIAL ASSISTANT *Deborah Sommers*

SPECIAL PROJECTS ASSISTANT EDITOR *Charles G. Frohman*

VICE PRESIDENT/GROUP DIRECTOR *Regina Starr Ridley*

PUBLISHER *Peter Westerman*
pwesterman@mfi.com

ADVERTISING SALES
Yvonne Labat
(415) 905-2353
Stephen Nikkola
(415)905-2256
Christian O'Brien
(212) 626-2322
Claire Bright
(415) 905-2544

MARKETING MANAGER *Susan McDonald*

MARKETING GRAPHIC DESIGNER *Shari Flack*

ADVERTISING PRODUCTION COORDINATOR *Glenn Sadin*

CIRCULATION DIRECTOR *John Rockwell*

CIRCULATION MANAGER *Stephanie Blake*

SUBSCRIPTION INQUIRIES
U.S.: (800) WANT-OS2
Foreign: (708) 647-5960

Miller Freeman
A United News & Media publication

CHAIRMAN/CEO *Marshall W. Freeman*
EXECUTIVE VICE PRESIDENT/COO *Thomas L. Kemp*
SENIOR VICE PRESIDENT/CFO *Warren "Andy" Ambrose*
SENIOR VICE PRESIDENT *David Nussbaum*
SENIOR VICE PRESIDENT *H. Verne Packer*
SENIOR VICE PRESIDENT *Donald A. Pazour*
SENIOR VICE PRESIDENT *Wini D. Ragus*
VICE PRESIDENT/PRODUCTION *Andrew A. Mickus*
VICE PRESIDENT/CIRCULATION *Jerry Okabe*

BPA INTERNATIONAL

PRINTED IN THE U.S.A.

Editor's Comments

By DICK CONKLIN

Serving Our Clients

Client/server development has been one of our most popular themes since we started. And why shouldn't it? Downsizing is the theme of the '90s, and many a mainframe has yielded to a distributed network of desktop PCs and servers. There's no better C/S platform than OS/2. The popularity of client/server applications is also reflected in the best-selling computer books list. Our favorite examples are the books from Bob Orfali and Dan Harkey (you saw them here first, folks!). Bob and Dan (and Bob's wife Jeri) have an amazing talent for making complicated subjects easy to understand. If it takes little Martian cartoon characters to get a point across, that's fine with me!

In this issue, we once again welcome Bob and Dan back to *OS/2 Developer*. You guessed it—here's an excerpt from yet another new book—their *Essential Distributed Objects Survival Guide*.

While we're at it, let's welcome Steve Krantz to the ranks of client/server authors. Steve's article is taken from his new book, *Real World Client/Server*, an interesting commentary on one large corporation's migration to a C/S platform. The company? None other than IBM itself.

This issue completes our seventh year of publishing this magazine. In 1996, you'll see some big changes, but no letup in the flow of practical articles about OS/2 application development. As always, we greatly appreciate your feedback, your suggestions, your subscriptions, and of course your support. Don't be shy—we like to hear from you!

Dick Conklin

Dick Conklin

Contact	CompuServe	Internet
Dick Conklin (Editorial)	76711,1005, or OS2DF2 Forum	os2mag @vnet.ibm.com
Miller Freeman (Circulation)	76001,2317	mjohnson @mfi.com
Peter Westerman (Publisher)	76307,1054	pwesterman @mfi.com
Theresa Mori (Managing Editor)		tmori @mfi.com

OS/2 Developer: Vol. 7, No. 6, November/December 1995

Subscription information: U.S.: One year (six issues), $39.95. Canada, Mexico, and international surface mail, add $16 per year for postage. Canadian GST no. 124513185. International air mail, add $30 per year for postage. Foreign orders must be accompanied by payments in U.S. funds. For new orders and customer service, call (800) WANT-OS2 (800-926-8672) or (708) 647-5960 (fax: 708-647-5972). IBM employees and branch office customers can subscribe to *OS/2 Developer* through IBM Mechanicsburg's Systems Library Services (SLSS) using *OS/2 Developer*'s order number: G362-0001. POSTMASTER: Please send address changes to *OS/2 Developer*, P.O. Box 1079, Skokie, IL 60076-8079. Allow eight weeks for subscriptions to begin. *OS/2 Developer* (ISSN 1073-0729) is published bimonthly by Miller Freeman Inc., 600 Harrison Street, San Francisco, CA 94107, (415) 905-2200. Application to mail at second-class postage rates is pending at San Francisco, CA, and additional mailing offices.

© Copyright 1995 by Miller Freeman Inc.

OS/2 Toolbox

This issue, our OS/2 Toolbox column reviews CCC/Manager 3.0.1d.

By GUY SCHARF

CCC/Manager for OS/2

Guy Scharf

Configuration management consists of much more than simple version control and being able to recover prior versions. Softool Corp. has released CCC/Manager for OS/2 to address these needs. I reviewed the recently released version 3.0.1d.

CCC/Manager provides traditional version control features, with change and component management to allow access to historical versions of any file. But it also provides much more. *Virtual* configurations define multiple views of a project for different stages of a project, test, or production area, as well as any other view that you define. *Packages* group changes from multiple modules to a single unit that can now be migrated from development to test, release, or production status. *Auditing* and *Access Control* functions allow you to monitor activity and to ensure integrity of your software. CCC/Manager's strength is its ability to manage change in a complex environment, more than it is version control per se.

CCC/Manager stores version and configuration information in a database that by default reflects the directory structure of the project. While the database need not reflect the directory tree used for the project's source code, the default paths presented for different operations will be more useful if it does. When managing multiple products, each should be in its own database to use CCC/Manager's release mechanisms effectively.

CCC/Manager is licensed for the number of users who will be logging into it. Concurrent licensing is available for larger installations.

USING CCC/MANAGER

I installed CCC/Manager from two disks. Installation proceeded normally, except that the installation program demanded write access to the installation disk to update a DLL file. A 39-character license key, with a mixture of upper- and lower-case letters and special characters, has to be typed in during installation. CCC/Manager uses InstallSHIELD, with its well-known inability to handle long lines in CONFIG.SYS. The installation program detects this situation and advises you to change PATH and LIBPATH manually. In my testing, I did not have to to make these changes at all.

My first glimpse of CCC/Manager was inauspicious. I opened the demo database and was presented with three garbled windows. CCC/Manager's main windows are implemented as dialogs with a menu and sizing border. This approach results in controls that overwrite the title bar and border if the dialog is sized smaller than its controls. Fortunately, this is easy to fix: simply resize the windows so that all the controls are visible within the windows' borders, and then don't adjust the window sizes again.

Each time you start CCC/Manager, you must login to the database with your user name and password, the fully qualified path to the database, your working directory, and, optionally, the name of a configuration file. While CCC/Manager does not remember this information from one use of the program to the next, it does allow all of it to be specified on the command line. You can easily create a program object on the desktop, or a command file, with the required information passed as parameters.

CCC/Manager has four main windows and a session log. From the *File* window, you can check in files and administer users and the database (Figure 1). Figure 2 shows

the *Item* window from which you can check out files, manage locks, create reports, and compare and merge files. The *Configurations* window lets you create and manage releases and phases (Figure 3). From the *Packages* window, you can define packages of changed modules to group changed modules together for migration and promotion. The *Session* log shows the result of most operations (Figure 4).

Directories are shown in a tree view, using a custom CCC/Manager control. This control is more compact than a container, and it works well.

When you begin using CCC/Manager for a project, first define your users and groups of users. Access control is extensive; more than 30 functions can be allowed or disallowed for a user working on a specific configuration. An audit log is optionally maintained, and you can specify which of six classes of events are to be logged.

The power of CCC/Manager comes with its configurations that you create to represent a product life cycle. You must create a baseline configuration and load the application into the database for that configuration. Creating the baseline is easiest if your product is contained within a single directory tree. When you load the product into the database, CCC/Manager records the current status of all modules and defines that as a baseline. Versions of files are recorded as forward deltas to reduce check-in time. With the baseline established, you can define other configurations that are derived from the baseline. For example, you might have separate configurations for development, test, and production versions of the product. It is easy to define these configurations and their relationships to each other.

Checking out and checking in of files is straightforward. You check files out using the items window and check them back in from the file window. By default, files are locked exclusively so that only one person may have a file checked out. You may change a database to use concurrent locking instead of exclusive locking. With concurrent locking, several people may check out the same file. When you check the file back in, CCC/Manager will inform you if changes have been made since you checked it out. A merge tool will merge your changes with any other changes and then

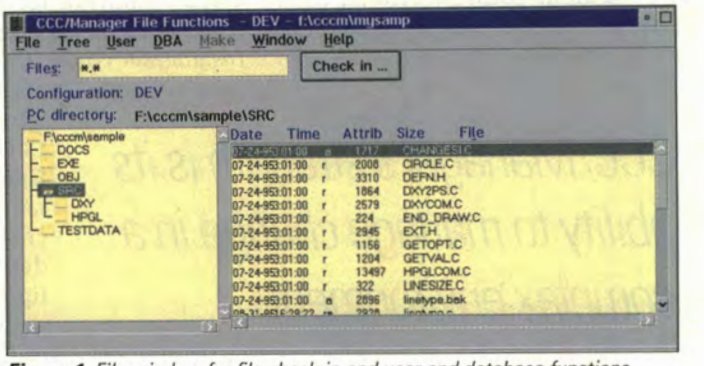


Figure 1. *File window for file check-in and user and database functions.*

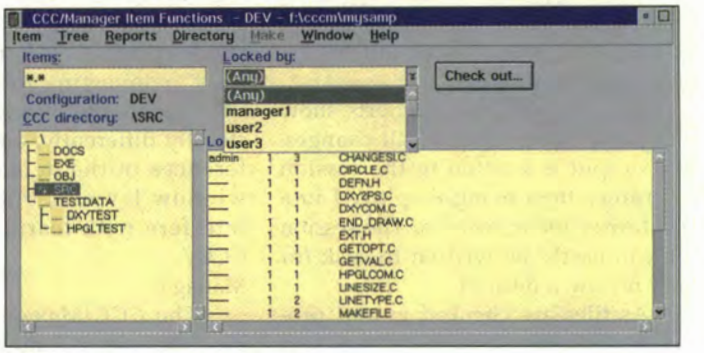


Figure 2*. Item window for file check-out.*

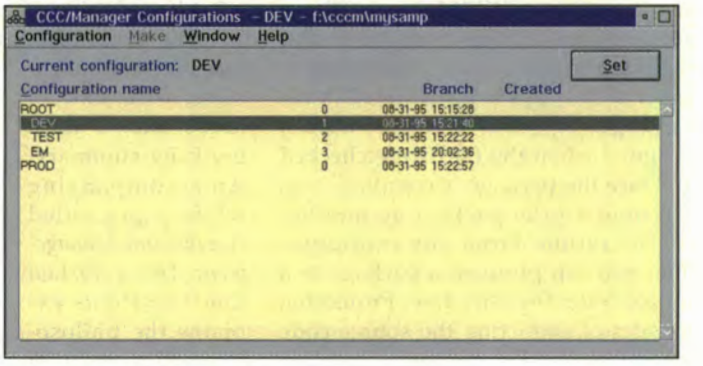


Figure 3. *Configuration window.*

Figure 4. *Session log.*

give you the opportunity to review the combined file and edit any conflicts. CCC/Manager also supports version branching and duplicate directories as ways to accomplish concurrent development. For long-term concurrent development projects, the better solution may be to define two development configurations, each derived from the same base configuration. These two configurations can then be merged to create a common, merged configuration for integration testing and later promotion to test and production status.

CCC/Manager's strength is its ability to manage change in a complex environment.

Version and *Change* reports show a summary and detail of all changes. The output is written to the session log rather than being displayed in a window or file of its own. The session log can easily be written to disk for later review if desired.

As files are checked in and project development progresses, you will eventually want to move some or all changes from the development to the test directory. CCC/Manager allows you to group changes to multiple files together in a package. You simply select all the files to be included in the package by the change names assigned when the files were checked in. Once the package is complete, you can migrate the package to another configuration. From any configuration, you can promote a package to a source code directory tree. Promoting consists of extracting the source code for the files in the package and placing it in the target directory tree. From there, you can recompile the configuration and confirm that everything still works.

Eventually, you have a new release. You define the new release as a new baseline configuration and create new development and test configurations derived from it. Any pending changes from the prior development and test configurations are merged forward into the new configurations, which replace the old ones for continued development.

You might also want a configuration for emergency fixes to the production version. It is easy to merge changes from the emergency fix configuration back into the development configuration so that they are integrated into the mainline code.

DOCUMENTATION

Documentation for CCC/Manager consists of the 14-page *CCC/Manager Installation Guide*, the 80-page *CCC/Manager Primer*, and the 290-page *CCC/Manager User's Guide*. The *Primer* takes you through creating a project, defining new configurations, checking in and checking out changes, promoting versions, and merging versions. While the *Primer* is written for the Windows product, I had little problem "mentally" converting to the equivalent OS/2 windows. The icons are named slightly differently, and there are differences in the default actions and window layouts, but these did not interfere with learning how to use CCC/Manager.

The *CCC/Manager User's Guide* describes every function of the product in detail. I found the manual to be clearly written, and it was easy to find what I was looking for.

The documentation explains configuration management well and in more detail than my brief summary. An accompanying white paper, titled *Application Management: Using Virtual Configurations* explains the philosophy and application of configurations in more detail.

CCC/Manager provides well-thought-out functions for managing versions and releases.

WARTS AND BLEMISHES

The documentation states that CCC/Manager's windows are designed to CUA standards. It would have been better if this wasn't stated—the windows violate CUA in many respects. Interestingly, the windows shown in the manual comply better with CUA than do the windows of the actual product. The problems are many; for example, mnemonics are used inappropriately, and multiple selection doesn't always do what might be expected. It appears that Softool intended the windows to be resizable, with the listboxes within the windows displaying more or less information. But this was not implemented, and the sizing border was left enabled. It would have been better to have disabled the sizing border, if Softool wasn't going to implement window sizing. These problems are irritating but do not seriously affect usability. Softool has said that it is addressing these issues.

When you check in or check out a file, or perform other operations, you must examine the session log to determine if the operation was successful. For most operations, no error window is displayed or bell sounded if the operation fails; you must watch the session log constantly. This approach is more reminiscent of exception reports from mainframe configuration management applications than it is of a good graphical user interface.

The check-in window would be simpler if a filter was used to show just unchecked-in files. Without a filter, you have to examine the entire directory to locate the files that are to be checked in. Both of these design issues will contribute to user error.

Don't close the Session Log window. I couldn't get it back without stopping and restarting CCC/Manager. It's OK to minimize this window, though.

CCC/Manager will optionally insert the revision history or eight other pieces of information in the file as it is checked out. However, the comment delimiters for the revision history are defined for the database, not for the file extension. Thus, it may be difficult to include revision history on files that require a different comment delimiter than is used for most of the files in a project.

Most windows are not refreshed automatically to reflect changes in

directories or files. It would be more convenient if the lists in the windows were always maintained in their current form, even if changes were made outside of CCC/Manager.

LIMITS OF APPLICABILITY

CCC/Manager is designed for the company that wants to manage its products and control changes. It supports multiple releases of products, as many phases (such as development, test, production) as your organization might want, client-customized versions of products, and emergency fix configurations. Virtual configurations are the strength of CCC/Manager and can be applied to solve many configuration management problems.

If you are looking for a programmer's version control tool, you probably won't like CCC/Manager. While command-line programs for check-in and check-out are provided, they are DOS programs requiring special DOS Settings. It may not be easy to integrate those tools with your editor or other development tools.

TECHNICAL SUPPORT

Technical support is available by telephone and e-mail. Softool has also created World Wide Web pages for a configuration management information center. Information articles are available on basic configuration management principles, tips for implementing configuration management in an organization, and lifecycle and process management. The configuration management web page is located at http://www.softool.com/CMCenter.html.

SUMMARY

CCC/Manager will appeal more to organizations that need to manage multiple versions of products or are ready to adopt a formal change management strategy than to organizations needing only simple version control. I like CCC/Manager for managing projects. It provides well-thought-out functions for managing versions and releases as well as batching changes. The availability of the product on diverse platforms should be attractive to installations managing projects that span several platforms.

As a programmer, I would like better integration with my editors and other development tools to increase flexibility in checking out and checking in modules on an ad hoc basis during development. The user interface could be much improved by careful attention to Presentation Manager details and CUA guidelines. Softool has said that an upgrade to address user interface issues and to add OS/2 command line utilities will be released soon. I am looking forward to seeing these changes.

Guy Scharf is president of Software Architects Inc., a software development firm specializing in developing OS/2 Presentation Manager applications for vendors and business. He can be reached at CompuServe 76702,557 or at Software Architects Inc., 2163 Jardin Dr., Mountain View, Calif.

PRODUCT INFORMATION

CCC/Manager for OS/2.....$495
Annual maintenance.....$85

Vendor: Softool Corp.
340 S. Kellogg Ave.
Goleta, Calif. 93117
Phone: (805) 683-5777
Fax: (805) 683-4105
Internet: info@softool.com

Circle Reader Service Number 4

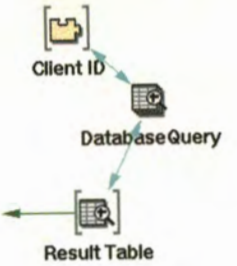
You know how applications look.

You know how to click a mouse.

You know how to draw a line.

You know how to watch

a computer do all the work.



This concludes your training in OO programming with VisualAge.

No one's debating the benefits of object-oriented programming. The only question is whether it's worth the time and money it would cost to implement.

With VisualAge™, the question may be irrelevant. Because its simplicity can easily remove the barriers between you and the fast development of object-oriented applications.

Can your software do this?

VisualAge is light years beyond mere GUI builders. It's a graphical environment that takes you through the entire process, from interface design to working application. As *InfoWorld* puts it, "a masterpiece of visual programming."

With the C++ edition, you work with "parts" from IBM's Open Class Library, creating visual links between them. They're easy to modify and compliant with standards, so they can be used across platforms, from PCs to the biggest servers.

When your project is complete, you've created an application with industry-standard code (C++ or Smalltalk). And in a fraction of the traditional development time, you're ready to deploy a true object-oriented application, with solid components that can easily be used over and over in future projects.

Of course, your full OO solution requires even more. That's why IBM offers more OO products, consulting, education and services than any other software company. To quickly take advantage of OO technology, call 1 800 IBM-3333 ext. GA 070 or visit http://www.software.ibm.com. You'll find that you've been in training for VisualAge all your life.

IBM

Solutions for a small planet™

In Canada, please call 1 800 565-SW4U. Outside North America, please contact your local IBM office. The IBM home page is located at http://www.ibm.com. IBM is a registered trademark and VisualAge and Solutions for a small planet are trademarks of the International Business Machines Corporation. All other company and/or product names are trademarks or registered trademarks of their respective companies. © 1995 IBM Corporation. All rights reserved.

Circle Reader Service Number 5

GUI Corner

This is the first in a series of articles, which uncover the hidden details of bitmaps and their manipulation.

By MARK BENGE and MATT SMITH

What's In a Bitmap?

Mark Benge

Matt Smith

Everyone is affected by the bitmap, yet most of us really don't understand how they work or are constructed. Various bitmap formats are defined within OS/2 to the extent that they seem convoluted and complicated. To compound things, there is a form of bitmaps that is effectively an array of images.

Over the next few issues, we will jump paws-first into bitmaps to try to learn how they are defined, how we can display them directly from bitmap files, and how to manipulate them and perform some really neat tricks with them. Along the way, we will try to explain some of the more mysterious details of bitmaps, such as ROPs.

So, where do we begin? How about we look to the origin of bitmaps?

A QUICK HISTORY LESSON

In the beginning, there were two types of displays used for interactive graphics. One was the random-scan vector display. This display type can be thought of in a similar fashion to that of the plotter. Using vector-type commands, it would draw lines and arcs on a cathode-ray tube. You would only need to give the display a starting point and an ending point, and it would draw the line.

The second type, the raster-scan display, became the more prevalent type. The raster-scan display can be thought of in a similar fashion to the dot-matrix printer. Information is recorded on a dot-by-dot basis. These dots are commonly known as pixels.

The major difference between the two display types is that the vector display contains only the information regarding the starting and ending points of lines and characters with the middle or void areas being ignored, whereas the raster display is responsible for each element (point) that constitutes each line or character. Therefore, much more information needs to be recorded by the raster display than by the vector display.

So, why the raster vs. the vector? Two real reasons. The need for a realistic display of images played heavily in favor of the raster display since it could display a three-dimensional image with greater ease than the vector display. The second reason was cost. Initially, the high cost of raster displays was due to the large amount of memory required because each addressable pixel required memory to record its state. A larger amount of memory was necessary to store the information within a raster display than a vector display. However, the costs dropped, and the raster scan display prevailed.

INTO THE HERE AND NOW

What is a bitmap? Well, as the name suggests, it's a map of bits. Yeah, right...

Just like a geographic map, each relevant item within that map describes a location. Los Angeles has a location of 34 0'N 118 10'W, whereas London is 51 30'N 0 5'W. You cannot transpose these two locations.

Bitmaps are similar. A pixel, which is defined for location, cannot be transposed with a different pixel in a different location. If you do transpose them, the meaning of each would be different.

Bitmaps as we know and use them today really just describe a color at a particular location. When the colors are combined, we interpret them through the mechanics of the eye and then apply meaning to what we see.

BITMAP CONCEPTS

Essentially, the bitmap is a record of the color of a particular pixel position. With raster displays, this is 2, 16, 256, or 65,536 (64K) colors.

Many different schemes can be created

to record this pixel, and, essentially, the bitmap under OS/2 is one of those schemes. To make things more difficult, how do you ensure that a given bitmap can work on the different display types, with different capabilities and resolutions?

DEVICE INDEPENDENCE

The structures that are used within OS/2 to describe bitmaps have been designed with as much device independence as possible. During the days of OS/2 1.1, there was a simple bitmap definition. This definition defined the bitmap for a given device, which meant you had to create different bitmaps for each of the resolutions handled. Then, you had to save these bitmaps within the applications' resources using different IDs. Finally, at runtime, you had to determine the resolution you were dealing with and then pick the appropriate bitmap image from the applications' resources. Seemed like a lot of work.

Beginning with OS/2 1.2, the design goal of the bitmap was device independence so that one bitmap could contain different images of the bitmap, thus allowing the image that was appropriate for a given device to be selected.

Now you know why the bitmap appears to be so difficult. Instead of creating two, three, or four images of a bitmap for a set of different resolutions, you could create one bitmap (known as a bitmap array) that contained the different images for the different resolutions. Then, inside your PM application, you would only need to select the one bitmap through `GpiLoadBitmap`, and the system would figure out which of the images in the bitmap array would suit the particular display on which the application was running.

And to make life more fun, the structures that were introduced in OS/2 1.2 were revised and expanded for OS/2 2.0. (All of the necessary structures for both OS/2 1.x and 2.x are contained within the PMBITMAP.H header with the 2.x structures ending with a 2.)

Part of these changes were supposed to make it easier to create device-independent bitmaps, but some changes really are not very effective.

THE STRUCTURE OF A BITMAP

The simplest OS/2 bitmap consists of a structure, BITMAPFILEHEADER or BITMAPFILEHEADER2, a color table, and the image data. Listing 1 shows the structure values for the bitmap image of Figure 1.

Basically, the structure describes the bitmap through a series of fields. The most important fields to pay attention to are contained in the `BITMAPINFOHEADER` or `BITMAPINFOHEADER2` portion of the structure. The cx, cy, and cBitCount fields are used to describe the width and height of the bitmap and the number of colors in it. The cBitCount field defines the number of bits defined by each pixel position within the image data.

The following table depicts the number of colors as defined by cBitCount:

```
BITMAPFILEHEADER2 bfh2:
 USHORT  usType  = BFT_BMAP
 ULONG   cbSize  = 78
 SHORT   xHotspot = 25
 SHORT   yHotspot = 25
 ULONG   offBits  = 142
 BITMAPINFOHEADER2 bfh2.bmp2:
  ULONG cbFix      = 64
  ULONG cx         = 50
  ULONG cy         = 50
  USHORT cPlanes     = 1
  USHORT cBitCount    = 4
  ULONG ulCompression  = BCA_UNCOMP
  ULONG cbImage      = 0
  ULONG cxResolution  = 0
  ULONG cyResolution  = 0
  ULONG cclrUsed     = 0
  USHORT usUnits     = BRU_METRIC
  USHORT usReserved   = 0
  USHORT usRecording   = BRA_BOTTOMUP
  USHORT usRendering   = BRH_NOTHALFTONED
  ULONG cSize1      = 0
  ULONG cSize2      = 0
  ULONG ulColorEncoding = BCE_RGB
  ULONG ulIdentifier  = 0
   argbColor[ 0]     = 0x00000000
   argbColor[ 1]     = 0x00000080
   argbColor[ 2]     = 0x00008000
   argbColor[ 3]     = 0x00008080
   argbColor[ 4]     = 0x00800000
   argbColor[ 5]     = 0x00800080
   argbColor[ 6]     = 0x00808000
   argbColor[ 7]     = 0x00808080
   argbColor[ 8]     = 0x00cccccc
   argbColor[ 9]     = 0x000000ff
   argbColor[ 10]     = 0x0000ff00
   argbColor[ 11]     = 0x0000ffff
   argbColor[ 12]     = 0x00ff0000
   argbColor[ 13]     = 0x00ff00ff
   argbColor[ 14]     = 0x00ffff00
   argbColor[ 15]     = 0x00ffffff
```

Listing 1. *Bitmap structure values.*

Figure 1. *Sample bitmap image.*

cBitCount	colors
2	2
4	16
8	256
16	65,536

The easiest way to determine the number of colors is to take the value of the cBitCount and rotate it left 16 positions:

```
ccolors = bfh2.bmp2.cBitCount << 16
```

If the number of colors for the bitmap are 256 or less, an RGB color table resides immediately after the BITMAPFILEHEADER or BITMAPFILEHEADER2 structure. It will contain either 2, 16, or 256 entries depending on the cBitCount value.

If the bitmap is a 64-K color bitmap, each item within the bitmap data is an RGB value.

So why the difference? Quite simply, because space is required to record the image. And, consequently, this is another area that makes the decoding of the actual bitmap data interesting.

The easiest (and also most difficult) way to understand this is to look at the actual image data for the various color types supported. Here we will look at a three by three image in 2, 16, and 256 colors as well as 64-K color representations (Figure 2). To conserve space and also make the examples easier, we are using the OS/2 1.x structure definitions. Each square in Figure 2 represents a pixel.

As you look at each of the structure definitions, you will really only notice a few changes between each of them. The offBits element will differ as a result of the color table between the structure and the image data. The cBitCount field will contain the appropriate value for the bitmap defined. Other than that, the values are the same within the structures.

As would be expected, the color table information will be different for each of the different bitmaps. And finally, the image data itself will be different.

For the two-color bitmap, the header is:

```
BITMAPFILEHEADER bfh:
 USHORT  usType  = BFT_BMAP
 ULONG   cbSize  = 26
 SHORT   xHotspot = 1
 SHORT   yHotspot = 1
 ULONG   offBits  = 32
 BITMAPINFOHEADER bfh.bmp:
  ULONG cbFix   = 12
  USHORT cx     = 3
  USHORT cy     = 3
  USHORT cPlanes  = 1
  USHORT cBitCount = 1
   argbColor[ 0] = 0x00000000
   argbColor[ 1] = 0x00ffffff
```

The image data is:

```
A0000000040000000A0000000
```

For the 16-color bitmap, the header looks like:

```
BITMAPFILEHEADER bfh:
 USHORT  usType  = BFT_BMAP
 ULONG   cbSize  = 26
 SHORT   xHotspot = 1
 SHORT   yHotspot = 1
 ULONG   offBits  = 74
 BITMAPINFOHEADER bfh.bmp:
  ULONG cbFix   = 12
  USHORT cx     = 3
  USHORT cy     = 3
  USHORT cPlanes  = 1
  USHORT cBitCount = 4
   argbColor[ 0] = 0x00000000
   argbColor[ 1] = 0x00000080
   argbColor[ 2] = 0x00008000
   argbColor[ 3] = 0x00008080
   argbColor[ 4] = 0x00800000
   argbColor[ 5] = 0x00800080
   argbColor[ 6] = 0x00808000
   argbColor[ 7] = 0x00808080
   argbColor[ 8] = 0x00cccccc
   argbColor[ 9] = 0x000000ff
   argbColor[ 10] = 0x0000ff00
   argbColor[ 11] = 0x0000ffff
   argbColor[ 12] = 0x00ff0000
   argbColor[ 13] = 0x00ff00ff
   argbColor[ 14] = 0x00ffff00
   argbColor[ 15] = 0x00ffffff
```

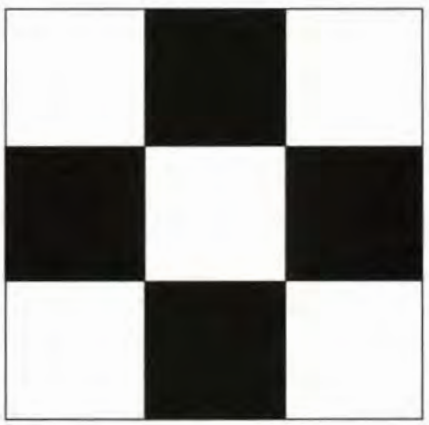


Figure 2. *Three by three bitmap image.*

and the image data:

```
F0F000000F000000F0F00000
```

The 256-color bitmap header looks like (there are 256 items for the color table, only a few are shown here):

```
BITMAPARRAYFILEHEADER :
 USHORT  usType  = BFT_BITMAPARRAY
 USHORT  cbSize  = 40
 USHORT  offNext  = 0
 USHORT  cxDisplay = 1024
 USHORT  cyDisplay = 768
 BITMAPFILEHEADER bfh:
  USHORT  usType  = BFT_BMAP
  ULONG   cbSize  = 26
  SHORT   xHotspot = 1
  SHORT   yHotspot = 1
  ULONG   offBits  = 908
  BITMAPINFOHEADER bfh.bmp:
   ULONG cbFix   = 12
   USHORT cx     = 3
   USHORT cy     = 3
   USHORT cPlanes  = 1
   USHORT cBitCount = 8
   argbColor[ 0] = 0x00000000
   argbColor[ 1] = 0x00000080
   argbColor[ 2] = 0x00009200
   ...argbColor[254] = 0x00f7f7f7
   argbColor[255] = 0x00ffffff

0F000F00000F00000F000F00
```

Finally, for the 64-K color bitmap:

```
BITMAPARRAYFILEHEADER :
 USHORT  usType  = BFT_BITMAPARRAY
 USHORT  cbSize  = 40
 USHORT  offNext  = 0
 USHORT  cxDisplay = 0
 USHORT  cyDisplay = 0
 BITMAPFILEHEADER bfh:
  USHORT  usType  = BFT_BMAP
  ULONG   cbSize  = 26
  SHORT   xHotspot = 1
  SHORT   yHotspot = 1
  ULONG   offBits  = 140
  BITMAPINFOHEADER bfh.bmp:
   ULONG cbFix   = 12
   USHORT cx     = 3
   USHORT cy     = 3
   USHORT cPlanes  = 1
   USHORT cBitCount = 24
```

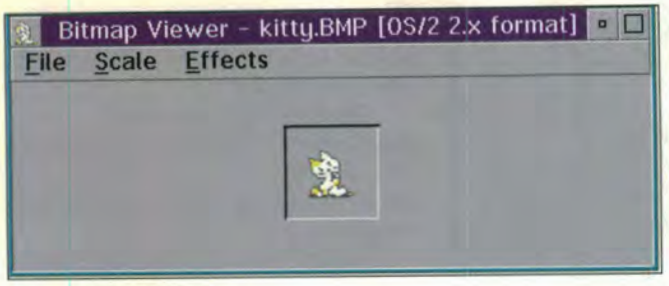


Figure 3. *Bitmap viewer.*

and the image data:

```
FFFFFF000000FFFFFF000000
000000FFFFFF000000000000
FFFFFF000000FFFFFF000000
```

So what's the big deal? The magic of completing the bitmaps' interpretation lies within the image data itself. What the data represents (except for the 64-K color bitmaps) is an index into the RGB color table for the pel position.

Consider a 16-color bitmap. It will have a color table of 16 elements. Therefore, if we had:

```
5 7 9
1 3 8
6 4 2
```

the first pel would be RGB[5] and the second pel would be RGB[7].

This is the concept; the trick lies in the actual representation within the bitmap. What this really means is that depending on the number of colors, the index position is defined differently.

For the two-color bitmap, the image data is (we are showing it in row format to make it easier to see how the data corresponds to the image):

```
A0000000
40000000
A0000000
```

When you look at this, you'll find it difficult to understand the data. Each row (or scan line) within the bitmap data (and this applies to all bitmap data) must begin on a 32-bit (`ULONG`) boundary.

A two-color bitmap index is very simple. There are only two index values, 0 and 1. Therefore, each of the bits for the two-color bitmap consists of a series of bit index values. By changing the above to binary, you get for each of the rows:

```
10100000000000000000000000000000
01000000000000000000000000000000
10100000000000000000000000000000
```

See the pattern? Each byte would then contain eight pel index values.

The 16-color bitmap operates in a fashion that is similar, only this time it looks like:

```
F0F00000
0F000000
F0F00000
```

Here, each RGB index is represented by a nibble (4-bits) instead of a bit.

A 256-color version of the bitmap looks this way:

```
0F000F00
000F0000
0F000F00
```

using each byte as the index to the RGB color table.

Finally, the 64-K bitmap represents each pel with the RGB color value (each RGB colors is 3-bytes):

```
FFFFFF000000FFFFFF000000
000000FFFFFF000000000000
FFFFFF000000FFFFFF000000
```

With this now under our belts, it is time to look at how we can use this information. How about a bitmap viewer?

A BITMAP VIEWER

One of the best ways to understand bitmaps is to build a viewer that can read in a .BMP file from disk and display it. Figure 3 shows the bitmap viewer in action.

Through the File Open menu item, you can pick a bitmap file to display. This file is then read in from the disk and interpreted (yes folks, the operative word here is interpreted).

Due to the history of bitmap evolution under OS/2, you'll need to structure the bitmap code to interpret the information that is being read. Because there are two different bitmap types, 1.x and 2.x, you'll need to interpret various header information to be able to know how to correctly reference the data contained.

Further, the bitmap can either be a single image or an array of images, which complicates things even more. Oh yeah, Windows bitmaps are similar, but, alas, the header information is different.

So, what have we done to handle this? Basically, we read in the entire image from the disk into a buffer of the appropriate size. Assuming the image is an OS/2 bitmap, it can start with one of two structures: `BITMAPARRAYFILEHEADER` or `BITMAPFILEHEADER` for OS/2 1.x type bitmaps, or `BITMAPARRAYFILEHEADER2` or `BITMAPFILEHEADER2` for OS/2 2.x bitmaps. The first element in each is `usType`. The second element is `cbSize`. It is these two elements that you should interrogate.

To determine if you are dealing with a `BITMAPARRAYFILEHEADERx`, check to see if `usType` is `BFT_BITMAPARRAY`, which if true means you need to interpret the data using the `BITMAPARRAYFILEHEADERx` structure. Each bitmap will be defined using this structure, and the means of walking through the structures from one to the next is by checking the value of `offNext`. When this value is 0, you have reached the last image within the array.

If the value contained within the `usType` element is `BFT_BMAP`, you are dealing with a `BITMAPFILEHEADERx` type of structure.

To determine whether or not the structures are 1.x or 2.x types, check the values of the cbSize elements. If you are dealing with a 1.x type of structure, this element will contain the value of

```
sizeof(BITMAPFILEHEADER)
```

By checking the size in cbSize, we can determine whether we should use the 1.x or 2.x structures.

What this really means is that we use similar code to interpret the information read in from the disk, except that one routine understands 1.x image information and the other understands 2.x image information.

STACKING THE ODDS

In our routines, we build a stack of images based on the bitmap array data. If the bitmap only contains one image, we only have one image within our stack. This will allow us to display whichever image is appropriate.

Figure 4 shows the bitmap viewer with the images from a bitmap array bitmap. It also shows the pop-up menu that is displayed when you click button 2 mouse over the image. This pop-up menu allows you to select the information to view for the image.

When you have selected one of the pop-up menu items, the information

Figure 4. *Bitmap viewer showing images from array.*

displayed, like that in Figure 5, is taken from the bitmap structures.

When you select the pop-up menu item for the color table, the View Color Table dialogue (Figure 6) will allow you to edit the color table entries.

SO WHICH IS THE RIGHT ONE?

A further dilemma revolves around the bitmap array. The question that is always asked is, "Which image do I pick?"

The trick with the bitmap is to know how to interpret the header information. For the bitmap array shown in Figure 4, the following depicts the format of the file:

```
image 1: 50 x 50 x 16 colors
BITMAPARRAYFILEHEADER2
BITMAPFILEHEADER2
BITMAPINFOHEADER2
RGB[16]
image 2: 25 x 25 x 16 colors
BITMAPARRAYFILEHEADER2
BITMAPFILEHEADER2
BITMAPINFOHEADER2
RGB[16]
image 3: 50 x 50 x 2 colors
BITMAPARRAYFILEHEADER2
BITMAPFILEHEADER2
BITMAPINFOHEADER2
RGB[2]
image 4: 62 x 62 x 256 colors
BITMAPARRAYFILEHEADER2
BITMAPFILEHEADER2
BITMAPINFOHEADER2
RGB[256]
image 1 data
image 2 data
image 3 data
image 4 data
```

In the bitmap viewer, you will notice that when you pick a bitmap initially, even if it is a bitmap array, only one image is selected.

The best way to explain this is to understand how and why the Icon Editor constructs bitmap arrays. The previous example depicts four images contained within the bitmap file. The first image—50 by 50, 16 color—is the default image. What this means is that if none of the other images are designed for the display type, this one is used. It is also supposed to correspond to a VGA type of resolution, or, as described by the Icon Editor, Independent Color Form (=VGA). This format is otherwise known as low res. The second image is a special-case image. It is described by the Icon Editor as Small Color Form. It is 25 by 25, 16 color, which is exactly 50% of the size of the Independent Color Form image.

The third image is defined by the Icon Editor as Independent BW Form (1.1). It is 50 by 50, 2 color—the same size as the default image but with only two colors.

Finally, the last image is defined by the Icon Editor as 8514. It is also known as high res. It is 62 by 62, 256 color, which is 125% larger than the default image, as it is intended for displays that are 1,024 by 768 or higher resolution.

Now what? How do we use this info to determine how to pick the correct bitmap? Look at the `BITMAPARRAYFILEHEADERx` fields `cxDisplay` and `cyDisplay`. If these fields are greater than 0, they will contain the resolution for which the image was intended. If the image is high res, the `cxDisplay` element will contain 1,024 and the `cyDisplay` element will contain 768. If the display on which you are presenting the image is at least this resolution, you would use this bitmap array image.

Alternately, if the image is a 2.x style image, you could try to use the cxResolution and cyResolution elements of the `BITMAPINFOHEADER2` to match display resolutions. This value, in theory (and the very operative word here

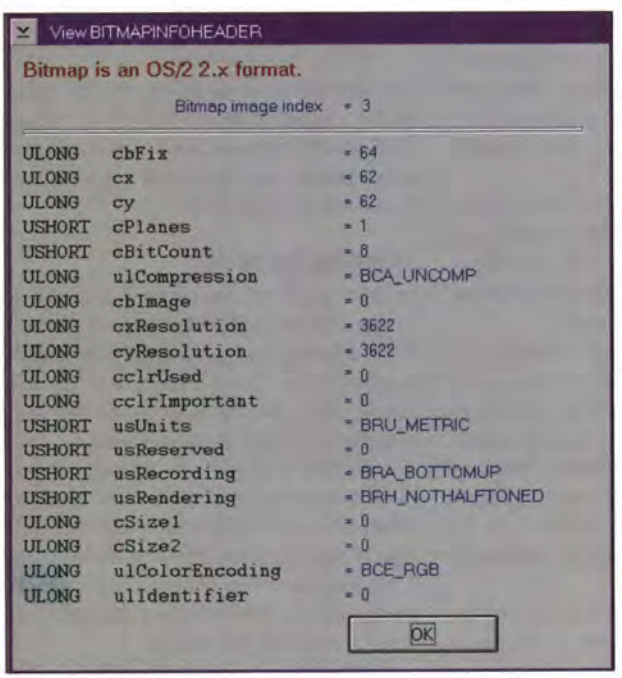


Figure 5. *Bitmap structure contents.*

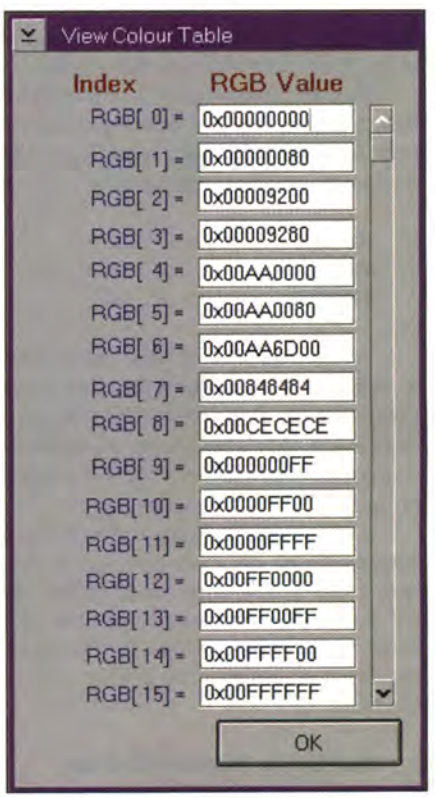
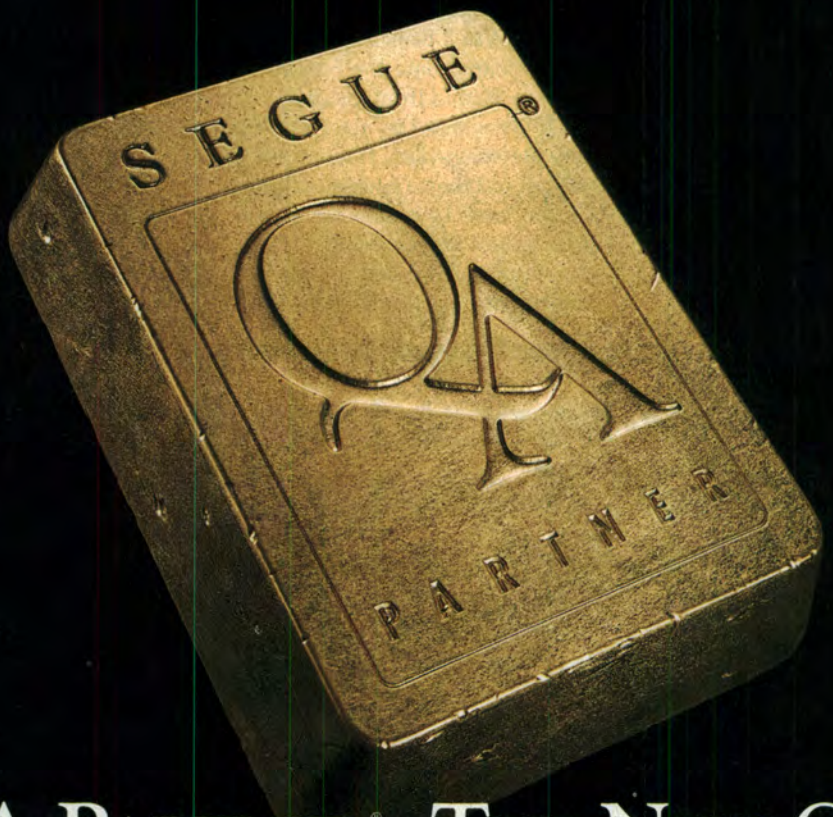


Figure 6. *View color table dialogue.*

• Windows • Windows NT AXP • Windows NT Intel • Windows95 • IBM OS/2 PM • IBM OS/2 Warp • Macintosh •

• Macintosh Power PC • Motif/DEC AXP OSF/1 • Motif/DEC AXP OpenVMS • Motif/DECstation ULTRIX • Motif/DEC VAXstation VAX VMS •

• Motif/Sun SPARC Solaris • Uniface • Gupta SQLWindows • Powersoft PowerBuilder • Oracle CDE • XVT • NCR • Motif/Sun Intel Solaris •



QA PARTNER®: THE NEW GOLD STANDARD IN SOFTWARE TESTING.

THE INTEGRATED QA TESTING PLATFORM — FOR *YOUR* PLATFORM

"QA Partner's language-based approach offers the best client/server testing solution on the market."
— Hurwitz Consulting Group, Inc.
December 1993

"Given our druthers, we would use QA Partner as our one in-house testing system."
— Tim Parker, *UNIX Review*
January 1995

"Honestly, I don't know how I would test this application without QA Partner..."
— Christine Comaford
PC Week, August 15, 1994

"...Segue's QA Partner. The hands-down winner when it comes to technical prowess."
— Tim Sylvester, *Client Server Today*
February 1995

Segue Software is the world's fastest-growing automated software testing company. No other testing product on the market empowers QA professionals like QA Partner. Call 800-922-3771 for further information on QA Partner today.

Segue
SOFTWARE, INC.

1320 Centre Street
Newton Centre, MA 02159
Internet: info@segue.com

QA Partner is a registered trademark of Segue Software, Inc.
All other system and product names are either trademarks or registered trademarks of their respective companies.

• VBX • Motif/IBM RS/6000 • Motif/SCO • Motif/SGI • Motif HP 9000/700 • Motif/Sun SPARC SUN OS •

Circle Reader Service Number 6

is theory), would be checked against the value contained in `CAPS_HORIZONTAL_RESOLUTION` and `CAPS_VERTICAL_RESOLUTION` from the `DevQueryCaps` API. Why theory? Well, simply put, no two different adaptors define the metrics in a similar fashion. Also, the display drivers may simply use a given model (that is, 1,024 by 768 while displaying in 1,280 by 1,024 resolution).

So, where does this leave us?—Low res and high res. Low res can be thought of as anything below 1,024 by 768 resolution and high res as being 1,024 by 768 and above.

THE REST OF THE VIEWER

What else does the viewer do? Well, in this version, it scales the bitmap images, displays device information, and allows you to flip the color table around. Why? Basically, to show you how to manipulate the bitmap once it has been read in from the disk.

If we wanted to, we could manipulate the image data. It wouldn't be too difficult to take an image and invert it. All you have to remember is how the index values are recorded.

As an added bonus, the bitmap viewer can read in a Windows bitmap and display it as though it were an OS/2 bitmap, even though the header information for Windows bitmaps are different.

NEXT TIME

Next time, we will discuss that thorny issue of ROPs. They really are quite simple once you understand the basics.

Mark Benge, *IBM Software Solutions, Cary, N.C., is a staff programmer who joined IBM in 1989 and has worked on various CUA '91 controls for OS/2 2.x. He works in IBM user open class library development, where he was involved in the implementation of C++ classes for drag-and-drop in C Set++ 2.1. Mark has a B.S. in computer science from Western Carolina University. He can be reached on CompuServe at 73532,2063 and via Internet at banzai@vnet.ibm.com.*

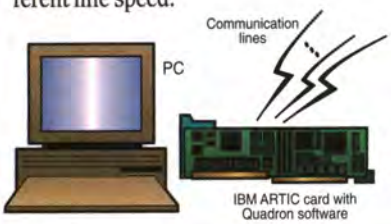
Matt Smith, *Prominare Inc., Toronto, Ont., is lead architect for the Prominare Development System, an OS/2 2.x advanced GUI development environment. Matt has been actively involved with OS/2 since 1988 and cofounded Prominare in 1990. He has a degree in architecture from the University of Waterloo and can be reached on CompuServe at 70363,1175 or via Internet at msmith@prominare.com.*

Quadron development tools for IBM ARTIC cards:

How to communicate better.

Increase system efficiency.
Our development software allows you to shift communication tasks from a PC to an IBM ARTIC co-processor card specially designed to handle them.

You'll gain CPU independence and multiple high-speed ports. Plus, each port on the IBM card can potentially support a different protocol and a different line speed.



Choose just what you need.
We have off-the-shelf solutions for your everyday data transmission needs, and tools for custom situations. You can select from HDLC/SDLC, Bisync, Async, X.25, LAP-B and custom protocols.

Your host PC can be an ISA, EISA, or MCA machine running various operating systems. And possibly most important, you can port co-processor code that you write from one operating system to another with little or no modification.

Serve vertical markets.
Quadron and IBM have teamed up to help people worldwide manage demanding applications such as:

- Travel reservation
- Stock market data delivery
- Telecommunications switching
- Retail point-of-sale purchases
- Process control
- Automatic teller services
- Shop floor control.

Work simpler, work smarter.
Quadron software is easy to install, easy to use, and downright productive. The high-level, C-language API makes co-processor programming simpler. Our user manuals have numerous code examples to speed you on your way. And our support, should you need it, will quickly serve up accurate answers to questions, whether they come in by phone, fax, or through our BBS system.

Act right away.
For today's dependable solution to tomorrow's communications needs, contact us now.

Quadron
209 East Victoria Street
Santa Barbara, CA 93101 USA
fax 805-966-7630
telephone 805-966-6424

IBM Business Partner

© 1995 Quadron Service Corporation. All trademarks are the property of their owners.

REFERENCES

Credits: screen captures were done through OpenShutter from One UP Corp. and touched up through PaintBrush in Win-OS/2.

Special thanks to Michael Cooper for providing insight into bitmaps.

You can download Bmp1.Exe from these electronic sources:

CompuServe's OS2DF2 forum (*OS/2 Developer* section).

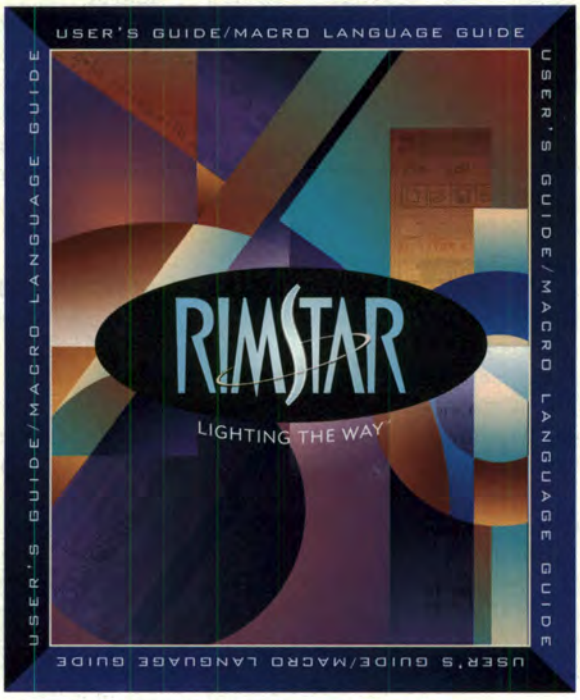
File area 11 on the IBM PCC BBS, (919) 517-0001, OS/2 Tools section on the OS/2 BBS for IBM TALKLink customers—ftp to the server address prominare.com and log in as anonymous. The source is located in the /pub/prominare/guicornercodecache subdirectory.

Also check out the new GUI Corner home page at www.prominare.com/prominare/guicorner.html.

Brief and to the point:

RIMSTAR was the first native OS/2 PM editor. In fact, it was the first GUI editor of its kind. So what's *new?*

Introducing RIMSTAR's 30-Day Free Trial offer. Right now, free of charge, you can use the leading OS/2 editor for 30 days – *fully functional* with our syntax coloring, complete ANSI 'C' macro language, smart C/C++ indenting and template expansion, no limits on line length, file size, number of files or windows, compile and jump to errors, hex editing, and source code browser for 'C'....



"My copy of Brief has been permanently retired! Keep up the good work." A.I.

For a limited time, RIMSTAR is yours – ***free*** – to use for 30 days.

But why talk about features when you can try out the leading OS/2 editor for yourself – ***free***? Call now for your copy: 1-800-746-7007. Outside the US, call 603-778-2500 or fax your order: 603-778-2408.

If you like it, you can have the RIMSTAR Programmer's Editor for OS/2 or Windows, Windows 95 and Windows NT for only $199.

RIMSTAR
LIGHTING THE WAY™

DESIGNED BY PROGRAMMERS FOR PROGRAMMERS

Circle Reader Service Number 8

Programming Insider

In this issue's Programming Insider, you will see how you can take advantage of the power of the workplace shell, use DLLs to reduce the amount of work in an application, and also minimize the amount of code you have to write and debug.

By DAVID E. REICH

Drag Me, Drop Me, Treat Me Like an Object

David Reich

In the classic programming and usage model, users start a program, open a data file, manipulate the data and then save and/or print the file. This is an *application-oriented* approach. The application is the center of the user's focus, having to be launched, fed, and maintained.

Object-oriented, on the other hand, is more than just putting objects in folders on the desktop and having objects represent your printers and other desktop tools. Object-oriented is also a way of thinking and a way of having users interact with the various parts of the computer.

In the application-oriented world, all you had to do was ensure that your application could be started from the shell, and you were done. Now, when users are as comfortable working with computer objects as they are with real objects, you need to change your way of thinking about the programs you write. You need to look at making your program a part of the computer system, not just something to be started and used.

To accomplish this, begin thinking about the documents or other objects that you want to manipulate (database records, files, graphs, or sets of data). Think of the objects as the operands and the application as the operator, to use mathematical terms. From now on, users no longer start applications, they open objects. The objects are simply presented to them in the way in which they are accustomed to seeing them. If it is a graph object, it is presented in their graphing program. If it is a database file, it is presented in their database program or transaction processor, or whatever program they use to manipulate their data.

This presents challenges in writing applications, and more precisely, object manipulators. These challenges arise in how you will use function and how you will architect it to be efficient and flexible.

In the past, I've written about not writing your application as an object but writing an object class (or classes) to represent the different types of data you might want to store. I've discussed the merits of reusable code and using DLLs for a purpose. This subject lends itself very well to such coding practices.

OBJECT-ORIENTED PRINTING

Let's first start with a straightforward subject: printing. Users of application-oriented, graphical software look to the menu bar for the File and Print menus. From there, one sees some dialog with which to select print destinations, paper orientation, and other attributes and then a push button that says Go, Print, or OK. The print job is on its way, as long as the user knows the printer name, some of its capabilities, and other tidbits of information that the dialog window might require. The user also needs to start the application, feed it a data file, and tell it where to put it. This is a lot of work to enable a user just to take an existing piece of data and put it on a printer.

In the object-oriented world, the user knows which folder holds the desired data. The user opens the folder, sees the object that represents this data, and drags and drops it on the printer. The user is then presented with a window that has some predefined values for paper orientation and such, and if no changes are desired, the user says OK and the job is off to the printer. The user never sees or knows about an application. The user only knows that (s)he wanted a piece of data printed, dragged and dropped it there, and it happened.

As I wrote last year in my two-parter on

"InSOMnia and Workplace Shell Programming" (May/June and July/August 1994), writing an object class to represent your documents or data files is not terribly involved. It does, however, require SOM programming and takes advantage of the Workplace Shell programming model. The point of this column is not to rehash what you've already seen but to expand on it.

To recap, your task is to enable drag-and-drop printing. The system (Workplace Shell) default behavior can only print plain text or printer-specific files, because OS/2 cannot, and should not, be expected to know about all of the data file formats that all of the available applications support. To support graphical printing through the device independence afforded by OS/2, the originator of the print job must be able to create a presentation space and a device context, and play or paint the data into the presentation space. To do that, the application must also know the data format so it can decipher it and draw it appropriately.

Given that the shell cannot be expected to know your applications' data format, you can force the user to open the application, open the data file and choose File, Print, and go through the application-oriented way of doing things. Or you can provide a way for the user to drag and drop a document on a printer and eliminate a lot of work. To accomplish this task (assuming you do not store your data file as plain text or printer-specific, in which case your work is already done), you will write an object class, most likely as a subclass of the Data File class (`WPDataFile`).

If all you want to do is enable drag-and-drop printing, you can override some basic methods to give yourself a type and an icon along with `wpPrintObject`. The `wpDataFile` class's behavior for `wpPrintObject` is to put up the "which type of file" dialog and then act accordingly, either creating a presentation space and device context and drawing the data, or passing it straight through with no translation (for printer-specific files.). You, however, will be overriding `wpPrintObject` and doing something different.

NOW FOR SOMETHING COMPLETELY DIFFERENT

When an object is dropped on a printer, it gets called at the `wpPrintObject` method. If the class that the object belongs to does not handle or override `wpPrintObject`, the method call is passed to the parent of that class, and so on up the chain until it hits a class that handles it. In the case of `WPDataFile`, that class handles the method call if the subclass does not. What you are doing is defining behavior different from the `WPDataFile` class's behavior for `wpPrintObject`.

When your object is called at `wpPrintObject`, it needs to take action to print the data object on the printer specified. First off, you are given the printer destination and a pointer to the object that is to be printed. In fact, the method call gives you the entire data structure needed to call `DevOpenDC` to create a printer device context. All you need to do is figure out how to take the object pointer and get the data from the object and interpret it.

To get the name of the data file that the dropped object represents, you can call `wpQueryRealName`. This will give you the file name for the file you need to print. Now you can see how to use the power of multithreading, code sharing, and the Workplace Shell.

At this point, you have been called at `wpPrintObject`, you are executing inside the code that you have written to override this method in your object class, and you have a print destination structure and a file name. You could simply call `DosExecPgm` or `DosStartSession` to start your application, pass in a flag such as `"/p"` to specify starting it up in invisible print mode, and pass in the print destination structure and the file name with code in the application to handle it—but that is the quick and dirty way. There is a much more elegant way to accomplish this, with much less overhead and much more reuse of code.

Starting a whole new session or process is a high overhead proposition. Most users are running in a memory-constrained system, no matter how much they have, and that much overhead just to print a file is very undesirable.

Rather than take you through the baby steps of a more elegant implementation, I'll just jump right into it. Use DLLs for your printing functions. DLLs are primarily used to share code between processes. You may say that your program is only one process. Remember, however, that your object classes run in the Workplace Shell's process.

If you take the "ugly" approach and start the application to print the file, you are

incurring all of the overhead associated with starting the process or session. By using DLLs for your print function, you can call it from the application when needed, or from another process such as the Workplace Shell process (and not load the code until or only if you need it). Let's look at the mechanics.

In a DLL, you write functions. It is not an executable program per se, but a library that contains functions and is callable by multiple processes. It is in this DLL that you can write the print handler. It will have one externalized entry point, such as `MyPrintFunction`. In this function you can accept the print destination structure you received from the `wpPrintObject` call, also with the real file name you received from `wpQueryRealName` in the `wpPrintObject` override in your class. This function will be called either from the application if it is running or from the object class that represents your data files. This is the consistent interface to your print function.

You must be aware of the effects you have on the rest of the user's computer when you add your code to the system.

Inside this function, you will create the printer device context and presentation space using the print destination structure provided, pull in the data file, interpret it, and draw it to the presentation space. You also have the option of calling `DevPostDeviceModes` for the printer to offer the user the option of changing some of the printer defaults.

Since this is your program, you can write it to interpret the data file format and work with it. You could also do this in the object class, but why have the code in the application and the object class, too? If you make a change to one, you must also change the other. Dual maintenance is something we try to avoid.

To enable all of this, simply export `MyPrintFunction` from the DLL, and create a .LIB file for the DLL. You will use this .LIB file to link the application as well as the object class to be able to call `MyPrintFunction`. Once there, a call to the function will resolve to your DLL and you can print without the overhead of another process or session—just a function call from your object class into some shareable application code.

WHAT ABOUT THREADS?

I'm sure by this time you're asking about threads. This is especially important when using this method of printing as opposed to doing it in the application, because when printing from the object using drag and drop, you are executing `MyPrintFunction` in the Workplace Shell's process, and, hence, using its threads. If you simply had single-threaded code in your application, shame on you. If you use the shell's thread and hold it for the duration of the print job, double-shame on you.

You could start a thread in the application before you call `MyPrintFunction` and just call it on this new thread, and you could do the same in the object class you write. This solves the problem. However, it again gives you double code. It would be easier and more efficient to call `DosCreateThread` right inside `MyPrintFunction`. You could then immediately return control to your caller and process the printing on this new thread.

Using this method, you create a thread only in one place and have only one set of code to maintain and debug. If you decide to write more applications, you can use this utility, or personal print library in all of them without having to code extra thread creation or other overhead each time.

FINE TUNING

There is one other bit of fine tuning I need to touch on before you begin writing the code to exploit these features. This should also give you some food for thought when you write anything to do with DLLs. Go back a little bit and reread the part about the .LIB files for your DLL. Notice anything interesting there?

Recall the fact about DLLs that have functions that other programs use and import by name using the .LIB file. The fix-up record in the executable code or other DLL is stored in the header of the EXE or DLL. Because when that module loads, the fix-ups must be performed. If you call `MyPrintFunction` by name in your object class, then whenever an object belonging to your class is opened (which will hopefully be often since you will not only write the class for printing, but for full object-oriented function with the application) the print DLL will be referenced and loaded if it is not already. This is more overhead than you need, especially if the user has no intention of printing the data object.

The function can be called by address, using `DosLoadModule` and rather than by name. By using this method, you delay loading the code until you really need it, rather than loading it just because the module that references it has been loaded.

There is nothing wrong with calling it by name and linking with the .LIB file when it comes to the application (if you still want to provide the user a File, Print menu option), but there's no reason to have the Workplace Shell load your DLL just because your document object class has been instantiated. These are the things you need to think about in coding this advanced function and how to gracefully cooperate with the system functions.

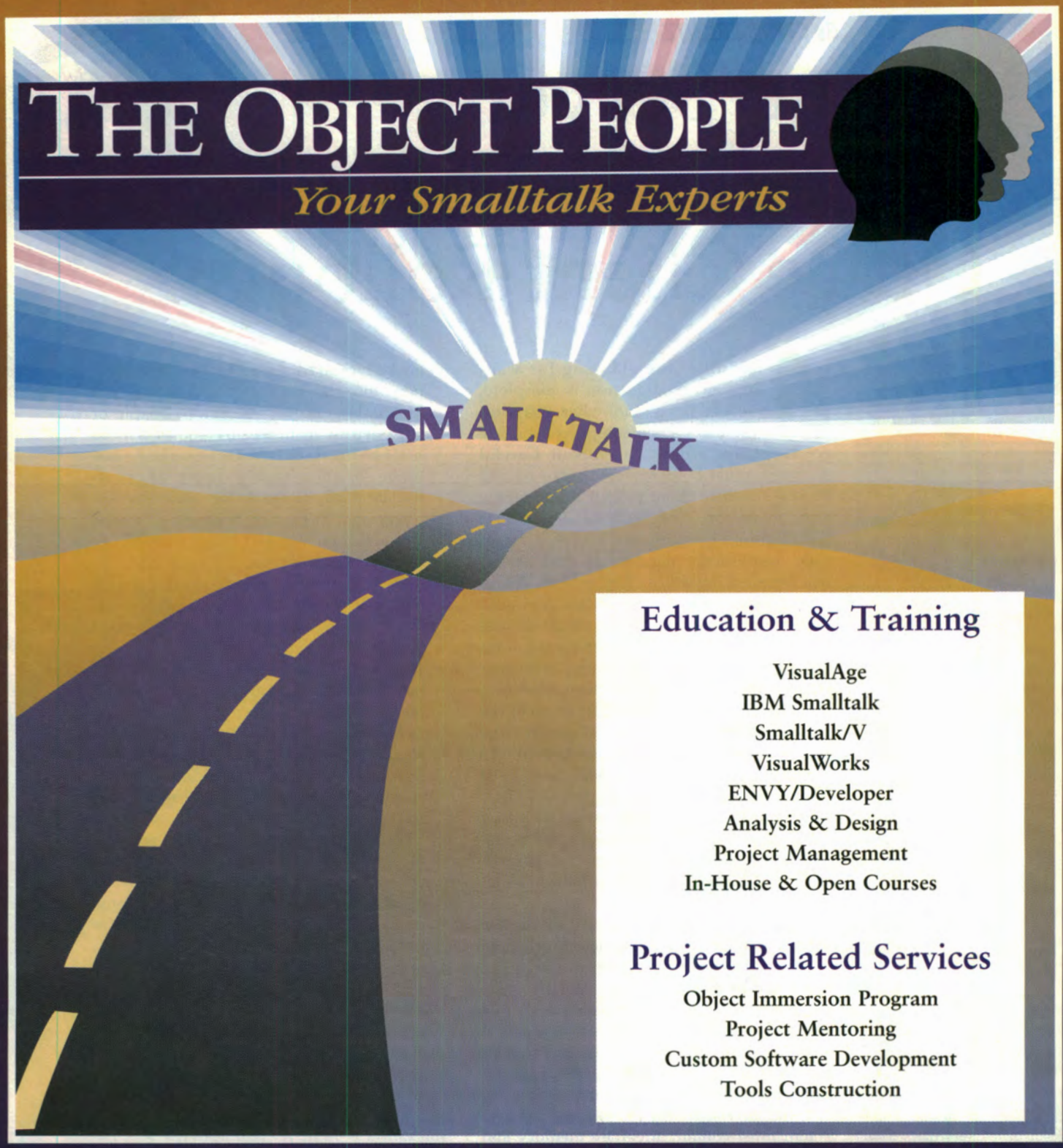
So, you've now seen a really concrete example of using DLLs for more than just sharing code among processes of your own application. This is just the tip of the iceberg in what you can do with the functions provided by OS/2. As you saw in the final tuning and thread tips, you must also take care in how you use the system resources and be aware of the effects you have on the rest of the user's computer when you add your code to the system. That is just what you will be doing in the new "seamless" world: making your application a part of the user's computer, not just something to start, stop, feed, and maintain.

David Reich *has been with the IBM OS/2 development team since 1987. He has worked on many parts of the system, supported customers and application developers, and traveled the world giving seminars and teaching OS/2. His latest book is* Designing High Performance OS/2 Warp Applications, *published by John Wiley and Sons. He can be reached on Compuserve at 76711,632 or via the Internet at speedracer@bocaraton.ibm.com.*

Distribution: "71743.452" 71743,452

TRANSITIONING TO SMALLTALK TECHNOLOGY?
INTRODUCING SMALLTALK TO YOUR ORGANIZATION?

Travel with the team that knows the way...



The Object People Inc. 509-885 Meadowlands Dr., Ottawa, Ontario, K2C 3N2
Telephone: (613) 225-8812 FAX: (613) 225-5943

Smalltalk/V is a registered trademark of Digitalk, Inc.
VisualWorks is a trademark of ParcPlace Systems Inc.

VisualAge and IBM Smalltalk are registered trademarks of IBM.
ENVY is a registered trademark of OTI Inc.

Circle Reader Service Number 9

During a client/server migration, there comes a time when you turn over the computers to your users and say "Go." Here's how to have a successful rollout.

By STEVE KRANTZ

Client/Server Migration—The Office Client Rollout

Steve Krantz

After all of the planning for a client/server migration is complete, there comes a point when you actually give the users some computers and say go. This is the office client rollout. Careful planning must be performed prior to and during an office client rollout to guarantee user satisfaction. This article details the IBM Boca Raton, Fl. experience during a large-scale client/server migration in 1993-1994. It will focus on the office client rollout, detailing requirements, the transition to client/server computing, and some critical success factors. Although this focuses on the Boca experience, most of the activities can be generalized and applied to your client rollout.

The office client probably impacts the bulk of the users and should receive the greatest attention throughout any client/server migration project. In Boca, over 2,700 computer systems fit this category. An office client is a PC with e-mail, calendar, phone directory, network connectivity software, and, optionally, word processing, spreadsheet, and business graphics applications installed in an office.

Within the client/server paradigm, the industry-standard PC is a generally accepted platform for the office client. As the office client will be used by most, if not all, office workers, professionals, and managers in a business, understanding its requirements is very important. Basic requirements for an office client are illustrated in Figure 1. Let's examine each element in the figure:

1. Industry-standard PC system. The system must contain adequate memory and disk storage to support the selected operating system and applications. An Intel 80486 processor operating at 33 MHz or higher, 8 to 16 megabytes (MB) of memory and a 200-MB hard disk should be considered a typical configuration for a full-function office client.
2. A color graphics monitor capable of supporting from 640 by 480 (VGA) to 1,024 by 768 (XGA, SVGA) graphics resolution. This is a widely supported industry standard, adequate for most user applications. Fifteen-inch diagonal screen size and noninterlaced image support are optional requirements.
3. A robust, multitasking operating system, able to support a broad range of office and nonoffice applications with good performance.
4. A Graphic User Interface (GUI), that has consistent human factors and a good performance.
5. Network connectivity able to provide adequate bandwidth for all office applications. This should be a 16-Mbps token-ring or 10-Mbps Ethernet LAN adapter for starters.
6. Peripherals support may be a requirement for a subset of users. Typical peripherals requiring support include CD-ROM (internal) or a personal laser printer. Some users requiring multimedia support may require a sound card with stereo speakers or a video card to display video clips.

USER INTERFACE

A user interface is a general term for the way the user interacts with a computer system to get it to perform useful work. Consistency and user-friendliness were the key user interface requirements during the migration period. A common GUI was the obvious choice as provided by OS/2 Presentation Manager. It offered a flexible windows application environment with consistent controls and rich customization capabilities.

This article is excerpted from Steve Krantz's new book, Real World Client/Server, *which details a major mainframe-to-client/server downsizing project at IBM. ISBN 0-9633214-7-1, IBM GA23-1216. Reprinted with permission, Maximum Press. Copies may be ordered via phone (800) 989-6733) or fax (615) 254-2408).*

OPERATING SYSTEM

It is important to select a single operating system for the office client that meets your requirements. If it is designed well, it will support multiple applications efficiently, allowing them to share the client system hardware resources (such as processor, memory, and input/output devices) without error or conflict. It should support the broadest set of industry-standard applications to give you the broadest latitude in application selection. It is also a complicated piece of software that requires occasional service—bug fixes, upgrades for new features or releases, or customization for new applications or system hardware. When a single operating system is selected, fewer support staff are required to assist users. Standardized maintenance procedures can be created, thereby minimizing cost.

From all angles, OS/2 was the best choice as Boca's standard client operating system. It is a robust, 32-bit, true multitasking software system that supports most of the PC-based applications available, including DOS, Windows, and OS/2 applications. Over two-thirds of the users were OS/2 literate and had early versions installed already. Not the least significant in the selection was that IBM Boca Raton is the home of OS/2 development. This led to key advantages in support, quality, and (most important) acceptance.

OS/2 provides an industrial-strength base for client/server migration. It is a fully architected operating system with an advanced object-oriented user interface available. In addition, OS/2 has crash and application protection.

HARDWARE

Selecting industry-standard hardware simplifies upgrades and maintenance. The broadest array of hardware upgrade and peripheral products are available. Since Boca Raton was the home of the IBM PC Co., almost all of the PCs were to be either IBM PS/2 or PS/ValuePoint systems.

One mistake that was made was allowing individual departments too much leeway in ordering specific PC systems. The diversity of systems, even within a single department, added difficulty to the PC manufacturing process.

The Intel 80486DX, operating at 33 MHz (million cycles per second) and with 16 MB of RAM (random access memory), was selected as the minimally capable processor complex based on informal benchmark testing performed in 1993. The bottom line was that application performance was too slow without this processor-memory combination. The minimum disk storage capacity of 240 MB was selected based on two factors. First, the standard applications plus utilities took up approximately 160 MB of disk storage, so a reasonable additional amount needed to be available for data and other applications. Second, several models in PC Co. inventory offered a 240-MB hard file.

Hard disks were logically partitioned in a standard manner (a physical disk is said to be logically partitioned when the operating system treats it as more than one disk, each with its own drive letter). The disk partitioning was based upon practical experience with PC-operating systems. A 100-MB space was allocated to the C drive to contain operating-system-related code. This included OS/2, IBM Communications Manager/2, LAN Requester, DOS Dual Boot, and LAN Station Manager. Isolating the operating system and its utilities into its own logical drive enabled update or replacement of this code with minimal impact to any user applications.

A VGA CRT display, with 640 by 480 resolution and up to 256 simultaneous colors, was deemed adequate graphics display resolution for the office client, although many systems were equipped with either XGA or SVGA resolution.

The standard applications used the NetBIOS protocol for local LAN communica-

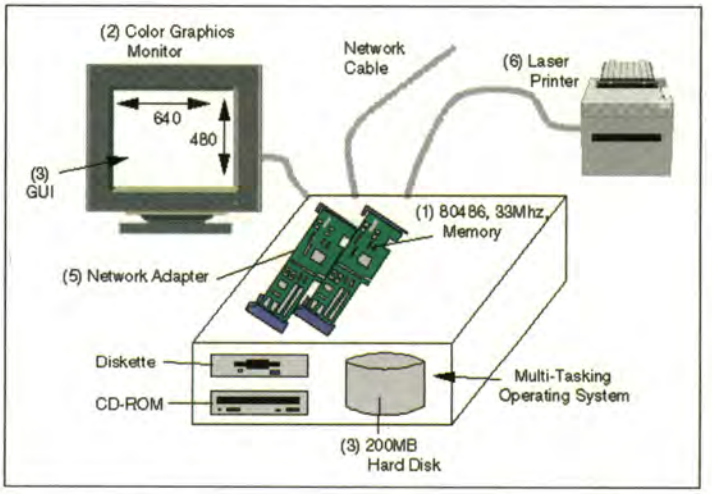


Figure 1. *Typical office client.*

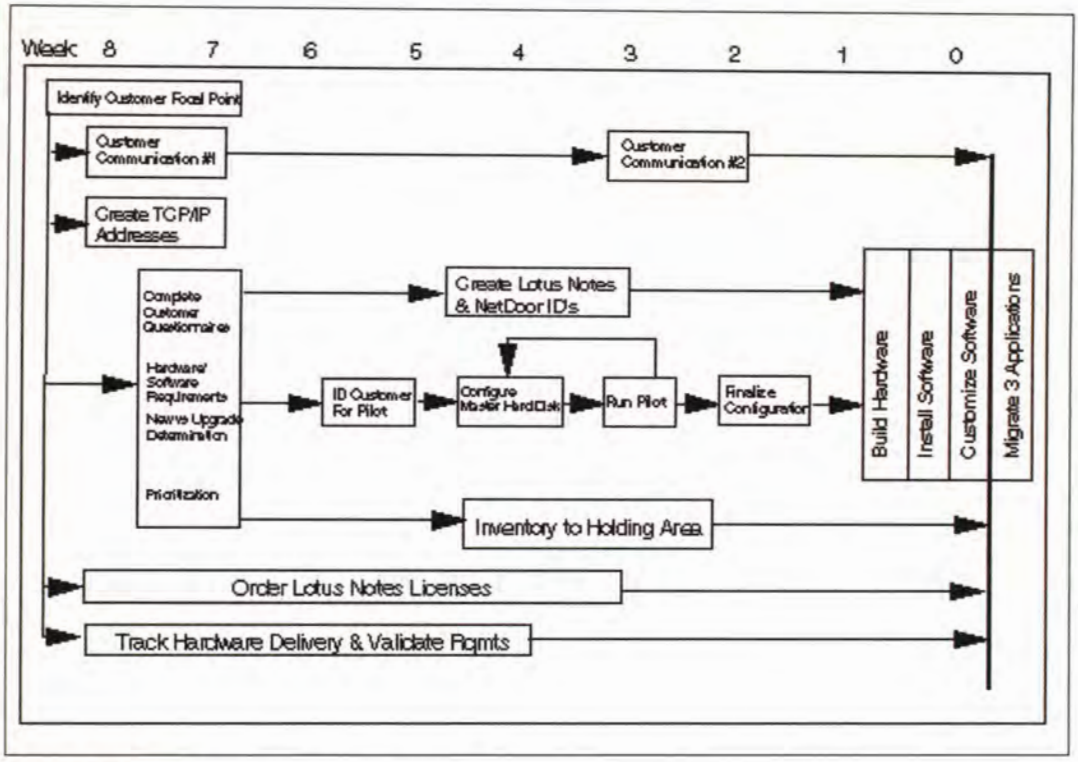


Figure 2. *IBM Boca Raton office client rollout process flow.*

tion. Some of the utilities used TCP/IP for both LAN and WAN communication. SNA, IBM's standard mainframe network architecture, was used for terminal emulation communication from PC to mainframe. The site network supported these multiple protocols transparently, as all the LAN internetworks were connected using bridges, which do not see the differences in the higher-level protocols.

The standard hardware interface to the site network was a token-ring adapter card, operating at 16 Mbps. All existing site PCs and workstations were already equipped with these adapter cards, and the site network was already operating at the 16-Mbps speed.

The most common peripheral in user offices was a personal printer, either a dot matrix or a laser printer. Host-attached laser printers supplemented this capability. A migration to LAN-attached printing is underway at the present time.

Some less common peripherals were CD-ROM drives and multimedia adapter cards (sound, video), which were increasing in number during the transition period.

OFFICE CLIENTS ROLLING OFF THE ASSEMBLY LINE

Once you have selected your standard office client's software and hardware, you face the daunting task of upgrading each user with the minimum system so that all can participate in the new environment. It requires a well-run team of trained specialists working closely with users and other support teams. For a large rollout, assembly line processes are called for to achieve success.

In January 1994 in Boca, over 2,700 office client PCs were to be either upgraded or replaced by the end of the third quarter of 1994 by the Client Rollout Team. Approximately, 700 systems were completed in 1993, so just over 2,000 remained in 1994. This meant that the target of installations in 1994 had to average 50 systems per week (or roughly 10 per working day) for the rollout to be successful. Based upon the inventory of installed PCs, the Client Rollout Team made the assumption that 70% of the systems were to be replacements (brand new systems) and 30% were to be upgrades.

The team started with the following general objectives, which are applicable to any client rollout:

1. Minimum disruption to user's current work.
2. All standard software preinstalled.
3. Final software customization would occur in the user's office by trained installer.
4. Installation would take no more than four hours.
5. All user data from current system would be migrated by the installer at time of installation.
6. Up to three applications from the former system would be migrated to the new system.
7. Users must return old PCs that had been replaced. The returned systems were to be reused or scrapped.

Your approach to client installation or upgrade should be comprehensive. It should be based on the Golden Rule (Do unto others as you would have others do unto you), because each office client upgrade has a significant psychological, intellectual, and physical impact on a new user. This was the approach used in Boca.

The Boca process was comprehensive and included the following steps:

1. Department focal points were identified to handle early information gathering and keep department members informed.
2. Area meeting presentations were performed to set expectations and communicate the plan.
3. Completion of a comprehensive user questionnaire, to establish user PC requirements, was required of every user.
4. Hard disks of new PCs were personalized with software through replication over the network from a "custom" master disk, which was for the department.
5. Personalized PC replacements or upgrades were carefully scheduled. This included installation of the standard desktop and maintenance of all prior capabilities (VM access, print, and up to three PC applications).

To support this ambitious effort, the Client Rollout Team set up a private local area network. It was used to efficiently transfer disk images of the standard applications software suite to fully configured PC systems.

The assignment of user focal points is highly recommended for your rollout. The focal point should be a person in each area or department to serve as your local agent to communi-

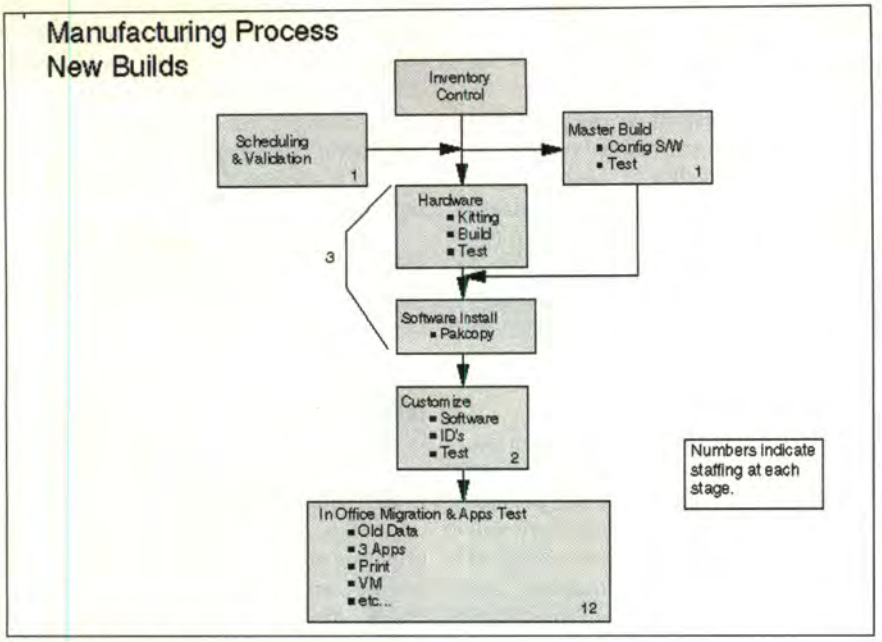


Figure 3. *Manufacturing process—new builds.*

cate directly with each person to receive a new or upgraded PC. The goal should be to have a focal point in each department (10 to 20 clients). At the IBM Boca Raton site, this proved to be unrealistic. A few organizations with 100 to 200 clients had assigned only a single individual to serve them. In addition, this was a part-time responsibility for the majority of the focal points. The focal point key duties should be:

1. Ensure completion of client questionnaires. The questionnaire should be taken online by most clients. It gathers comprehensive information of client requirements. Appendix D includes a sample copy.
2. Order network identifiers and key software licenses. At IBM Boca Raton, the Client Rollout Team ordered TCP/IP identifiers and Lotus Notes licenses for clients.
3. Track new PC orders and gather data on PCs to be upgraded.
4. Communicate with the Client Rollout Team and the clients throughout the rollout period. A weekly meeting run by the team should review all progress and problems. This proved to be a vital activity in IBM Boca Raton.

A well-defined group-by-group rollout process should be initiated once a focal point is identified. In Boca, the Client Rollout Team defined an eight-week rollout process for each group. It was viewed as a week-by-week countdown, where week eight was the kickoff leading to week one's eventual in-office installation (Figure 2). Week eight began with a presentation to the users (Customer Communication #1). This was an overview of the Client/Server Migration with specific emphasis on the client rollout. Other activities initiated during week eight were creation of TCP/IP addresses, ordering Lotus Notes licenses for each user, tracking of hardware deliveries, and validating their accuracy vs. requirements.

During week seven, emphasis was placed on completing user questionnaires. As these were accumulated, user requirements were solidified, in particular the breakdown of replacement PCs vs. PCs to be upgraded. This assessment was important in that it impacted installation resources. Through experience, the Client Rollout Team learned that complete PC replacements took from two to four hours in the user's office, whereas upgrades took from five to twelve hours. The accumulated user requirements were prioritized along with available inventory, as well. This resulted in transference of inventory meeting those requirements to a holding area awaiting final customization.

By week six, the user group's requirements were sufficiently solid to enable initiation of a pilot master build—the creation of a master hard disk to undergo testing during weeks five through two.

Figure 4. *Office clients ready to go.*

In week five, the Server Administration team created Lotus Notes and NetDoor identifiers for each user. These identifiers allowed users to access the applications.

In week three, each customer was contacted individually to set up a replacement or upgrade appointment. After the appointment was made, careful follow-up with the user was the rule to minimize no-shows.

At week two, the master hard disk configurations for an area were finalized. They included the standard applications plus any unique area requirements. Unique area requirements were specific applications that an area needed to do its job. For example, all OS/2 developers required an application which linked them to their software code library and problem-tracking database.

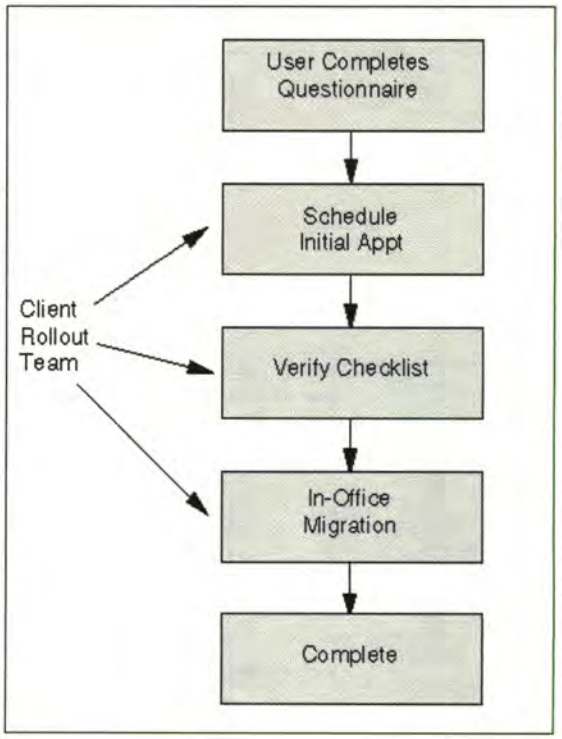


Figure 5. *Manufacturing process—upgrades (original process).*

During week one, the PC manufacturing process began. Figure 3 details the manufacturing process for new builds and identifies the staffing employed. Figure 4 shows a set of client systems ready to be delivered. The last step completed the manufacturing process with the in-office installation. Typical in-office installation time for new systems was two to four hours.

Figure 5 details the original manufacturing process for upgrades, which was completed entirely in the user's office. However, due to the variability of results with this method (from five to twelve hours), it was enhanced in June 1994 to combine in-office and manufacturing line sections (Figure 6).

Deliveries of new PCs or upgrades to existing ones received the personalized attention of a trained and skilled installer. The installers were contractors, specially trained to perform the installation or upgrade function. Every effort was made to provide same-day functionality for all user PC requirements. A complete information package was provided with the installation to answer most, if not all questions.

Up to three personal applications were migrated to new PCs by the installer, provided proof of license was presented and original diskettes were available. All old PC data were migrated to new PCs during the installation as well. This was accomplished by copying the entire existing hard disk files to a backup server on the network at the beginning of the installation/upgrade process. After installation/upgrade, the reverse copy was performed to an OLDDATA directory on the new or upgraded PC. If a PC was replaced, the old system remained in the user's office for about two weeks, just in case it was needed if the new PC failed. At the end of two weeks, the PC Returns department, a site services group, reclaimed the old system for either sale to another IBM site or refurbishment and redeployment.

The Lotus Notes application played a critical behind-the-scenes roll for the Client Manufacturing process. Eleven Notes databases were employed, serving as information repositories, workflow automation, and tracking applications. The Client Implementation Library Notes database contained procedure, process, and forms documents specific to the client rollout. The ISSC Document Library, another Notes database, contained general application installation procedure documents. The User Questionnaires were imported and maintained in a consolidated Notes database. (Imported means that the data required conversion to the Notes format. This was required because most users employed VM to complete the questionnaire.) The Master Configuration database contained descriptions of the hard file masters to be used by the Client Manufacturing process. Three Notes databases were used for client tracking and scheduling by the Installation Team. Three Notes databases were set up to inform users of their migration schedule, provide a migration checklist, and, most importantly, track purchase orders for new equipment.

The Boca eight-week client rollout process was highly successful. It can easily be generalized to apply to your client rollout.

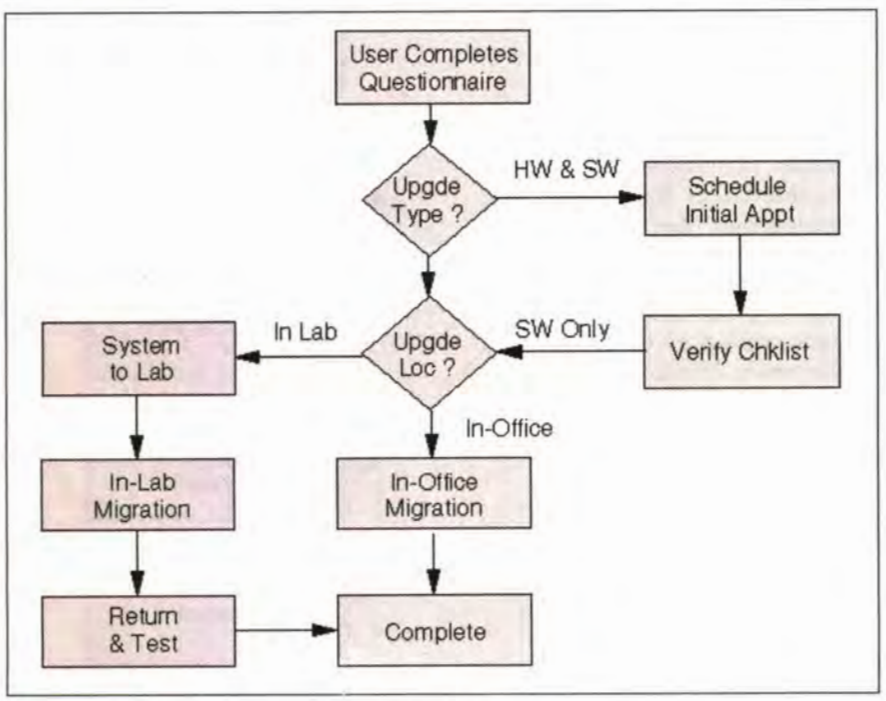


Figure 6. *Manufacturing process—upgrades (revised process).*

THE CHALLENGES OF A CLIENT ROLLOUT (KEEPING MURPHY AT BAY)

A client rollout is extremely challenging and requires a highly synchronized "manufacturing" process, worthy of a case study in industrial engineering! Many things can go wrong, and will (according to Murphy's Law), unless careful planning is performed. Several elements require synchronization, including:

- Gathering and logging customer data profiles and requirements. The comprehensive user questionnaires, when consolidated in a readily available database (for example, Lotus Notes), meet this requirement.
- Inventory delivery, staging, and control. Hardware delivery schedules can be unpredictable. They can be minimized through flexibility in schedule management, establishment of a three- to four-week inventory of readily accessible hardware, and supported by a (Lotus Notes) tracking database.
- Work-in-process management. Well-laid-out processes (see Figures 4 through 8) make the difference.
- Technical requirements changes. This is inevitable, as software levels change or last minute user requests filter in. The rollout team must be flexible and responsive to properly handle these changes.
- Workload balancing. The client installation team must be well-trained and proficient in their tasks to accommodate workload balancing. Installers should be PC-familiar at least, with application and some operating-system modification experience.
- Network-based software installation delivery. This is critical technology for large-scale client migration.
- Setting customer expectations. Customer expectation of each phase of the process should be appropriately set. A mass client migration takes a good deal of the "personal" out of a user's personal computer. This is increasingly inevitable as

client/server technology matures.

- Scheduling with the user. This is a delicate matter. Problems can be minimized through use of the airlines' method of overbooking (a key insight) and up-to-the-minute scheduling communication with the user.
- Additional user visits. Installation problems requiring additional user visits by installers add schedule pressure. This can be minimized through the use of carefully documented procedures and up-front testing by the installation team. The use of (Lotus Notes) tracking databases for sharing of information can help here.

By the end of the third quarter of 1994, the Boca Client Rollout Team rolled out over 2,000 new or upgraded PCs. They met the goal of over 2,700 enabled PCs delivered to the users.

CRITICAL SUCCESS FACTORS

The office client rollout is the heart of any client/server migration. Following are some key recommendations that will lead you to success in this critical phase:

- You should carefully identify the minimum hardware requirements for client PCs. This should be based upon performance evaluations of selected applications on a real system. Ensure that processor, memory, and disk storage can be upgraded if required. Secondarily, allow installation of peripherals as upgrades.
- There is no substitute for a complete inventory database of all user clients. This is required initially to scope the investment necessary to migrate to a client/server environment. It is required on an ongoing basis in order to track asset ownership and identify upgrade candidates. Investing in automatic client inventory maintenance is very worthwhile.
- You should create a completely defined process for a large-scale client PC rollout. This should include user communication (gaining buy-in, setting expectations), installation logistics, and tools. Flexibility in scheduling is also critical. Through careful schedule management and overbooking client PC installations, your rollout team can achieve success.
- Invest in groupware, groupware, groupware. The use of Lotus Notes databases enabled the incremental improvements made by the Boca client rollout team in their process.

REFERENCES

Krantz, A. S. *Real World Client/Server*, Gulf Breeze, Fl.: Maximum Press, 1995.

Tapscott, D. and A. Caston. *Paradigm Shift*, New York: McGraw-Hill Inc., 1993.

Steve Krantz *is a senior programmer with IBM in Boca Raton, Fl. He is the project leader responsible for the Boca Raton client/server migration in 1994. During his 27 years with IBM, Krantz has held several management and technical positions. He has a Ph.D. in Computer Science from Nova Southeastern University, an M.S. in Systems and Information Science from Syracuse University, and a B.A. in Mathematics from Queens College.*

GET THE LOWEST OS/2 PRICES-KWIK

VisualAge C++ for OS/2 version 3.0
by IBM

Version 3 of VisualAge C++ offers you visual programming tools that make building applications simpler.

3.5" disk with documentation	**$429.00** (MSR $525.00)	Order number 448
CD ROM	**$370.00** (MSR $449.00)	Order number 449
CD ROM with documentation	**$409.00** (MSR $489.00)	Order number 450

call for prices on upgrades from v2/v2.1

VX•REXX for OS/2
by Watcom

Powerful and easy-to-use integrated environment for developing OS/2 2.x PM applications, including a project management facility, visual designer and debugger. New to this version are Notebooks, Containers, Sliders, Popup menus, DDE, objects and more.

$95.00 (MSR $99.00)
Order number 211

SPF/PC v 4.0
by Command Technology

SPF/PC is a powerful file manager and full-screen text editor that emulates IBM's mainframe ISPF/PDF, providing a familiar environment for mainframe programmers faced with the challenge of developing on a PC.

$189.00 (MSR $295.00)
Order number 186

These products and many more PLUS our lowest price guarantee.

OS/2 Express promises to match or beat nationally advertised prices for OS/2 software we carry. Just place your next order with OS/2 Express. If you find a vendor that can fill your order for less elsewhere (including shipping, handling, credit card fees and other surcharges), send or fax us the ad within seven days. We'll match it.

O2 EXPRESS
1-800-OS2-KWIK

For orders call 1-800-OS2-KWIK (1-800-672-5945) • Fax (612) 823-6267 • International (612) 823-6255

Circle Reader Service Number 10

Writing a server from scratch using interprocess communications can be challenging. Here's an example that works.

By MICHAEL J. BARILLIER

OS/2 Server Design Using Named Pipes

Cutting-edge technology in client/server computing today would appear to be independent applications accessing a remote database. Scan through any computer-related magazine and count the number of database advertisements that do not include the term "client/server" somewhere in the text (from personal experience, this number should be very small).

Despite what the DBMS software developers claim, a database is nothing more than a resource, like memory or disk space, not an application server. While an application-to-DB server topology may be easy to implement, system data storage is left wide open to all clients, preventing any data hiding or encapsulation. All rules concerning the storage of the system's data must be built into each client, resulting in development and maintenance headaches. (While a database may start out normalized, it generally falls victim to entropy over time.) A server should be more than just a data repository—it should provide services and statistics, exert intelligent control over clients connected to it, and interact with clients in concepts relevant to the system, not just raw rows and columns. If a billing system is being implemented, transactions should consist of customer and invoice objects, not VARCHARs and INTEGERs.

Building a server isn't a trivial task, however. Management-types, obsessed with project completion dates, aren't very receptive when a developer announces that he/she is going to write a server from scratch. "What's wrong with using a database server?" they ask. "After all, every airline magazine article I've read on the subject says that's the way to go." Also, less-experienced programmers may be intimidated when faced with the challenge of interprocess communications. This article should help in two ways. First, it demonstrates basic server design and interprocess communication, and second, the sample code in Listings 1 through 4 can be gutted and used as a skeleton for many client/server environments.

SYSTEM OVERVIEW

The example server, Tserver, supports multiple simultaneous connections through named pipes. (The source code for Tserver is available on CompuServe in OS2DF2 as TSRVR.ZIP.) Many interprocess communications methods are available under OS/2 such as shared memory, named pipes, APPC and APPN, TCP/IP, IPX/SPX, NetBIOS and RS-232, to name a few. The only methods available on an out-of-the-box system, however, are shared memory, named pipes, and async. Of these, only named pipes is useful for interprocess communications between applications running either on the same workstation or on separate workstations. Client interaction with the server can be implemented using a simple `fopen/fputs/fgets/fclose` sequence. Also, named pipes can be accessed by DOS and Windows applications, given the proper software support, which means that the server can take advantage of OS/2's multitasking, crash protection, and system resources while serving clients running on lesser operating systems (yes, I am an OS/2 bigot).

Interaction between Tserver and clients is \n-terminated ASCII text messages, rather than the binary packets traditionally used in client/server communication. For example, if Tserver is sent the command string,

```
TIME
```

the client would receive something like

```
OK 13:21:05
```

I've found that text-based communications makes the server incredibly easy to access and test. One system that I built needed to have multiple clients simultaneously making different requests to stress test the system. Rather than write C applications, I wrote the clients in a few minutes in REXX, using lineout and linein to access the server. (More on client design later.)

SERVER DESIGN

Tserver is written as a multithreaded, VIO-based application. Interthread communication is through homemade message queues. These queues are simple singly linked lists with a mutex semaphore and an event semaphore to provide access serialization and message availability notification.

`main()`. The primary application thread is responsible for installing a break handler, starting the subthreads, dispatching messages between the threads, and updating the application's output, which consists of the current time and, whenever an interesting event occurs within the server, a descriptive text message. The message processing loop ends when a `breakhit` message (as posted by the break handler) is pulled from the main message queue.

Connect thread. The first thread started by the primary thread is the connect thread, which waits for clients to connect to an instance of Tserver's named pipe and posts the handle of the newly connected pipe to the main message queue. The function that implements this thread, `connectthread()`, is pseudocoded in Listing 1. `connectthread()` is loosely designed as a finite-state machine, with lots of labels and gotos. (While some programmers avoid gotos at all costs, communications applications in general are significantly simplified with their use.)

In general, `connectthread()` creates a named pipe, then waits on its shutdown semaphore for a short time. If not posted, `connectthread()` checks the pipe for a connection (the pipe is created in nonblocking mode). If a client has connected, the handle is passed to `main()` through `main()`'s message queue, a new instance of the pipe is created, and the loop starts over.

Although `DosCreateNPipe()` includes an

```
connectthread {

  // Clear the pipe handle (this is checked for a non-NULLHANDLE
  // value on thread exit ).

  hpipe = NULLHANDLE;

  // ... and fall through.

  create:    // Create a new instance of the pipe.

  hpipe = create NP_NOWAIT pipe using pipename and
     DosCreateNPipe();

  if (return code == ERROR_PIPE_BUSY)
   // Implies max pipes created.
     goto busy;

  connect:    // Check for client connection.

  call DosConnectNPipe() to determine if client connected;

  if ( no client connected )
     goto waiting;

  post a 'connected' message to main;

  goto create;

  waiting:    // Wait to see if the application is ready to exit.

  wait 250 ms on the shutdown semaphore;

  if ( shutdown semaphore posted during wait )
     goto completed;

  goto connect;

  busy:    // Pipe is busy.

  wait 250 ms on the shutdown semaphore;

  if ( shutdown semaphore posted during wait )
     goto completed;

  goto create;

  completed:    // Processing completed - clean up.

  if ( hpipe != NULLHANDLE )
     DosClose(hpipe);

  }
```

Listing 1. `connectthread()` *pseudocode.*

instance count that may be unlimited, the number of file handles available to an OS/2 process is limited by the FILES= statement in config.sys (the default value is 20). To avoid this limitation, the FILES= statement can be changed, but this would be system-wide and a waste of resources. An alternative is to check for an ERROR_TOO_MANY_OPEN_FILES return code from DosCreateNPipe(), and when found, to call DosSetRelMaxFH(), incrementing the number of handles available. To keep the handle maximum from growing without limit, Tserver sets the pipe instance count in DosCreateNPipe() to a reasonable value. Once the maximum pipe instances have been created, DosCreateNPipe() will return ERROR_PIPE_BUSY, and connectthread() will go back to waiting on the shutdown semaphore.

Because the number of handles is limited, the number of simultaneously connected clients is also limited. Tserver's design is transaction-based, so clients will not normally be connected for an extended amount of time. If the clients must keep their connections open for multiple transactions, and if a large number of clients exists, then the server design becomes more complicated.

Client handler thread. The third thread in Tserver is the client handler thread. As clients connect to Tserver's named pipe, connectthread() passes the handles to main(), which in turn passes them on to clienthandlerthread() through a message queue. clienthandlerthread(), pseudocoded in Listing 2, maintains a linked list of clientinfo structures, each of which contains the handle of the pipe to the client and a buffer containing text received from the client but not yet terminated by a newline.

New handles arriving in connected messages are packed into clientinfo structures and placed in the linked list. If a shutdownreq message is not received, then clienthandlerthread() works through the list and appends any data waiting in the client's pipe to the command buffer for that client. The command buffer is parsed and any commands executed. A \n-terminated result string will always be returned to the client following command execution, and the format of that string is dependent upon the command executed. The first token in the string will be "OK" or an error indicator such as "EBADCMD" or "EOFLOW". If the token is "OK", it will be followed by any data resulting from the execution of the command, such as the current time for the time command.

```
clienthandlerthread {

  cilist = NULL;    // Initialize clientinfo list

  waitq:    // Wait for input in the queue.

  wait 250 ms for message to arrive in queue;

  if ( no message received )
      goto checkclients;

  processq:

  while ( queue is not empty ) {

      msg = get next message from queue;

      switch ( msg->id ) {

          case connected:
              add a new clientinfo structure to cilist using the handle passed in
              the message;
              break;

          case shutdownreq:
              goto completed;

          }

      }

  checkclients:    // Check for incoming text in pipes.

  dataread = false;

  loop through cilist {

      read data from pipe into current clientinfo's buffer;

      // NOTE:  Pipe is in nonblocking mode, so DosRead may return ERROR_NO_DATA
```

Listing 2. clienthandlerthread() *pseudocode.* *(Continued on page 33.)*

CLIENT IMPLEMENTATION ISSUES

At the client end, a named pipe acts just like a file, and a named pipe handle can be used in DosWrite() and DosRead() calls just like any disk-based file handle. As mentioned before, a client can also access a named pipe using the C library file I/O function calls (for example, fopen, fputs, fgets). Listing 3 shows a series of calls to connect to Tserver and to request the current time using standard C file I/O (error handling has been omitted for simplicity). This example opens a locally named pipe, \pipe\time, but to access a remote named pipe the only change required is to place the server workstation name in front of the pipe name, for example, \\appsrvr\pipe\time. Note the use of fflush() between the fputs() and fgets() calls. The C Set++ documentation states that a file handle can be used for both reading and writing, but a call to a file-positioning function, such as fseek() or fflush(), must be made between read and write operations.

Access to a server using named pipes can be dropped into many different types of clients. Both OS/2 VIO (windowed and full-screen text processes) and Presentation Manager applications can access this type of server, although a PM application design should take into consideration interface responsiveness. Because server response time is generally unknown, the I/O sequence should not be placed

```
        if the client has not yet written to the pipe. This is not a read
        failure.

    if ( read failed ) {
        remove current clientinfo from cilist;
        close pipe for current client;
        continue;
        }

    if ( data was read ) {
     dataread = true;
        parse command buffer and process complete commands found;
        }

    move pointer to next clientinfo;

    }

// If no data was read ( all clients are quiet ), go to waitq.
// Otherwise, go to checkq, which only checks if messages are available and
    does not wait.  This is done under the assumption that if some data was
    read, there may be more.

if ( !dataread )
    goto waitq;

checkq:    // Check queue, but don't wait.

check message queue with 0ms timeout;

 if ( no messages available )
    goto checkclients;

goto processq;

completed:    -- Thread has completed and needs to clean up.

close any remaining clients in cilist;

}
```

Listing 2. `clienthandlerthread()` *pseudocode.* *(Continued from page 32.)*

in the PM message processing thread to prevent the client program from locking up during these transactions. DOS and Windows applications can also access an OS/2 server using named pipes—I have developed DOS applications that run on workstations connected using Novell LAN software with named pipe support, which access OS/2-based servers. I must admit, however, that I have not actually written a Windows application that uses named pipes, but because Windows runs on top of DOS (Windows isn't a true operating system, remember?), I have to believe that this design is possible.

Listing 4 shows an excerpt from `gettime.cmd`, which uses REXX to access the server. I generally use REXX clients when I want to knock out the client as quickly as possible, or I am building a client that performs some sort of background processing.

ENHANCEMENTS

Tserver is a very simple application, performing very few functions. What I consider the tougher part of the design—connection monitoring and client communications—can be used as is, so the largest part of the enhancement process will probably consist of modifying Tserver's command parser. The parser used within Tserver does nothing but use `stricmp()` to check the first word of the command buffer for either `"time"` or `"shutdown"`. A better solution would be to use an LALR parser, such as the code generated by Lex and YACC. Not only is the processing more efficient for large command sets or complex grammars, but the code will be far easier to maintain and expand.

Commands received by `clienthandlerthread()` are processed sequentially. Because of this, a command from a client, which takes a long time to process, will cause clients further down the list to wait. One solution to this problem is to use a worker thread pool. Once a \n-terminated command is detected, `clienthandlerthread()` would pull the `clientinfo` structure from the list, post it to the worker pool for processing, and then continue handling input from other clients. If there are waiting worker threads in the pool, the message would be posted to the waiting thread for immediate processing. If there are no waiting threads and the maximum number of threads has not been reached, then a new worker thread would be created. If the maximum number of threads had been created and all were busy, the command would be queued. The worker threads would check the queue when their assigned processing was completed. If no command was pending, they would go to a waiting state for an internally specified time and exit if no message was posted during that time. This avoids having threads wait for work when nothing has been available for a while. Once the worker thread completes its processing, the

```
char buffer[32];
FILE * f;

f = fopen("\\pipe\\time","r+");
fputs("time\n",f);
fflush(f);
fgets( buffer, sizeof(buffer), f );
fclose(f);
```

Listing 3. *Accessing Tserver from C.*

```
srvpnm = '\pipe\time'

call lineout srvpnm, 'time'

parse value linein(srvpnm) with
      retcode t

call lineout , 'Server time is' t

call lineout srvpnm
      /*  closes pipe  */
```

Listing 4. *Accessing Tserver from REXX.*

Take control of your OS/2 scheduling needs with
Advanced Task Scheduler for OS/2

ATS
Version 3.0

Are you looking for a way to manage your production job streams? Do you have programs that need to run after the completion of one or more other programs? Do you want to run programs on a regular basis? Do you want to process files that have just been received from another system or modified locally? Do you need to take action if a file is missing? Do you need to schedule around holidays and periods of heavy CPU load?

With ATS you can run your programs when YOU want them to run with out you having to be there.

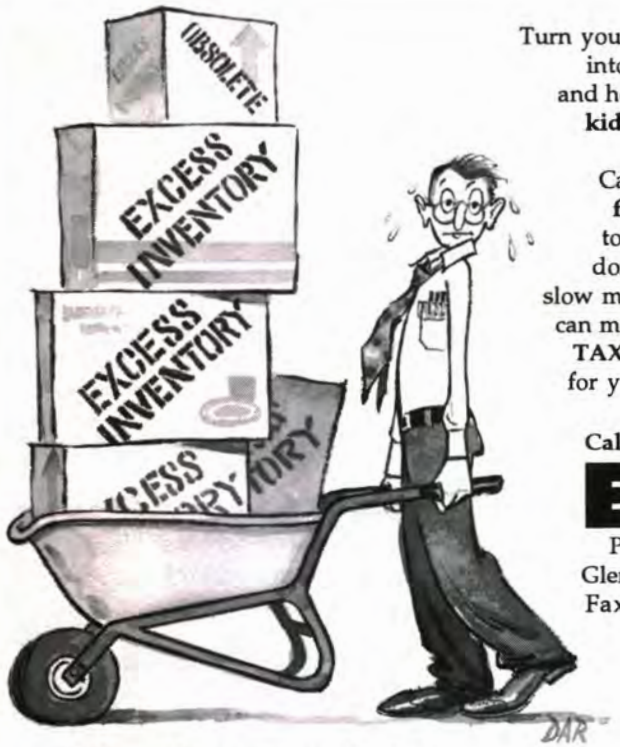
Upgrades Available from Version 2 and other Task Schedulers
Call or E-Mail for Details

Here are some of the features of ATS for OS/2:
* Manage Production Job Streams
* Programs can be scheduled to run anytime.
* Programs can be dependent upon Files, Other Programs, Holiday Schedules, and External Signals
* Integrated Operators Console
* Logging - To File and Console

New Features:
* Job Queues - for managing your resources while managing your tasks
* Periodic Scheduling
* COMPLETE API and Command Line Interface

Only $349

MHR
Software and Consulting
2227 U.S. Highway #1 *Suite 146
North Brunswick, NJ 08902
Voice: (908) 821 - 0359 * Fax: (908) 821- 0350 * CompuServe 70312,627



Turn your **excess inventory** into a tax break and help send needy **kids to college.**

Call for your **free guide** to learn how donating your slow moving inventory can mean a generous **TAX WRITE OFF** for your company.

Call (708) 690-0010

EAL

P. O. Box 3021
Glen Ellyn,IL 60138
Fax (708) 690-0565

Need some help with that?

Excess inventory today....student opportunity tomorrow

`clientinfo` structure would be returned to `clienthandlerthread()` and reinserted into the linked list.

And finally—a database. Nowhere in the server design presented in this article has a database been required, nor have the clients needed to access data using SQL. If the server will be maintaining a large amount of data, then a database should be used internally to facilitate its storage and access. However, the clients do not need direct access to the database, nor should they care whether the data is stored by the server in a true database, in flat files, or on stone tablets etched in Sanskrit. If, for some reason, the end users insist on using Query Manager, Paradox, or some similar product to build their own ad-hoc queries, then consider building a client support application to mirror system data to a separate user-accessible database. The clients can then build any query they like but will be unable to access the server's copy of the data and possibly crash the system.

SUMMARY

I had originally intended this article to be a soapbox tirade against the "client/server requires a database" philosophy, but decided instead that a viable alternative would be a better argument than uncontrolled ranting. The design, used by Tserver, is not technically advanced but is useable in many applications where traditionally a remote database server would be used. If nothing else, I hope this article will get at least a couple of developers (and managers) to break out of the old mindset and use their own creativity and programming skills.

Michael J. Barillier *is the head of Renegade Digital Technologies, a software development and consulting company in St. Louis, Mo. He has been working with OS/2 since 1989 and has extensive experience with client/server and object-oriented design. Barillier can be reached via Internet at mjb@inlink.com.*

An excerpt from Bob, Dan, and Jeri's latest book on client/server with distributed objects and components—including OpenDoc, CORBA, and OLE.

By ROBERT ORFALI, DAN HARKEY, AND JERI EDWARDS

When CORBA Objects Meet Transactions

Transactions are essential for building reliable distributed applications. So it should come as no surprise that transactions and distributed objects are getting married. The Object Management Group's (OMG) newly adopted Object Transaction Service (OTS) defines IDL interfaces that let multiple distributed objects on multiple CORBA ORBs participate in atomic transactions—even in the presence of catastrophic failure. OTS optionally supports nested transactions. The support of nesting and inter-ORB transactions provides an object foundation for dealing with the complex world of multistep consumer-to-business and business-to-business transactions.

THE CORBA OBJECT TRANSACTION SERVICE

The Object Transaction Service (OTS) is possibly the most important piece of middleware for distributed objects. OTS does a superb job of marrying transactions with objects at the ORB level. With OTS, ORBs provide a seamless environment for running mission-critical components. This feature alone gives ORBs a leg up over any competing form of client/server middleware. An ORB becomes the next-generation TP Monitor. In this article, we first go over what makes transactions so important. Then we introduce OTS.

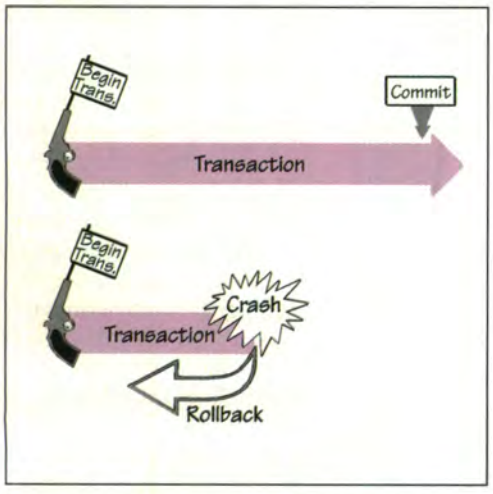


Figure 1. *The flat transaction: an all-or-nothing proposition.*

WHAT IS A TRANSACTION?

Transactions are more than just business events: They've become an application design philosophy that guarantees robustness in distributed systems. In an ORB environment, a transaction must be managed from its point of origin on the client, across one or more servers, and then back to the originating client. When a transaction ends, all parties involved agree as to whether it succeeded or failed. The transaction is the contract that binds the client to one or more servers. A transaction becomes the fundamental unit of recovery, consistency, and concurrency in a distributed object system. Of course, all participating objects must adhere to the transactional discipline; otherwise, a single faulty object can corrupt an entire system. In an ideal world, all distributed object interactions will be based on transactions.

Transaction models define when a transaction starts, when it ends, and what the appropriate units of recovery are in case of failure. The flat transaction model is the workhorse of the current generation of TP Monitors (and other transactional systems). It is called flat because all the work done within a transaction's boundaries is at the same level. The transaction starts with `begin_trans-`

Robert Orfali

Dan Harkey

Jeri Edwards

This article is excerpted with permission from The Essential Distributed Objects Survival Guide. *John Wiley & Sons, 1996, ISBN 0-471-12993-3. IBM publication number SR28-5898.*

action and ends with either a commit_transaction or abort_transaction (Figure 1). It's an all or nothing proposition—there's no way to commit or abort parts of a flat transaction. However, the newer transaction models—for example, nested transactions—provide a much finer granularity of control over the different threads that constitute a transaction. The newer transaction models are attractive because they have the potential to better mirror their real world counterparts.

OBJECT TRANSACTION SERVICE: FEATURES

CORBA's OTS provides the following features:

- *Supports flat and nested transactions.* All OTS implementations must support flat transactions; nested transaction support is optional. In a nested environment, the flat transaction is the top-level transaction.
- *Allows both ORB and non-ORB applications to participate in the same transaction.* OTS lets you interoperate object transactions with procedural transactions that adhere to the X/Open DTP standard.
- *Supports transactions that span across heterogeneous ORBs.* Objects on multiple ORBs can participate in a single transaction. In addition, a single ORB can support multiple transaction services.
- *Supports existing IDL interfaces.* A single interface supports both transactional and nontransactional implementations. To make an object transactional, you use an ordinary interface that inherits from an abstract OTS class. This approach avoids a combination explosion of IDL variants that differ only in their transaction characteristics.

OTS is a well-designed, low-overhead service that should perform at least as well as an X/Open-compliant procedural transaction service.

THE ELEMENTS OF THE OBJECT TRANSACTION SERVICE

Figure 3 shows the elements of OTS. Objects involved in a transaction can assume one of three roles: Transactional Clients, Transactional Servers, or Recoverable Servers. Let's go over these roles and see what they each do.

- *A transactional client* issues a set of method invocations that are bracketed by begin/end transaction demarcations. The calls within the bracket may be for both transactional and nontransactional objects. The ORB intercepts the begin call and directs it to the transaction service, which establishes a transaction context associated with the client thread. The client then issues method invocations on remote objects. The ORB implicitly tags the transaction context and propagates it in all subsequent communications among the participants in the transaction. The ORB also gets involved when the client issues a commit or rollback and notifies the transaction service. The client is oblivious to all this under-the-cover activity; it simply starts a transaction, issues its method invocations, and commits or rolls back the transaction.

What's a Nested Transaction?

Most of the alternatives to the flat transaction are based on mechanisms that extend the flow of control beyond the linear unit of work. Two of the most obvious ways to extend the flow of control are by chaining units of work in linear sequences of "mini" transactions (the chained transaction or Saga models) or by creating some kind of nested hierarchy of work (the nested transaction).

Nested transactions provide the ability to define transactions within other transactions. They do this by breaking a transaction into hierarchies of "subtransactions," very much like a program is made up of procedures. The main transaction starts the subtransactions, which behave as dependent transactions. A subtransaction can also start its own subtransactions, making the entire structure very recursive. A subtransaction's effects become permanent after it issues a local commit and all its ancestors commit. If a parent transaction aborts, all its descendent transactions abort—regardless of whether they issued local commits.

Figure 2 shows a main transaction that starts nested transactions, which behave as dependent transactions. Each subtransaction can issue a commit or rollback for its designated pieces of work. When a subtransaction commits, its results are only accessible to the parent that spawned it. The main benefit of nesting is that a failure in a subtransaction can be trapped and retried using an alternative method, still allowing the main transaction to succeed.

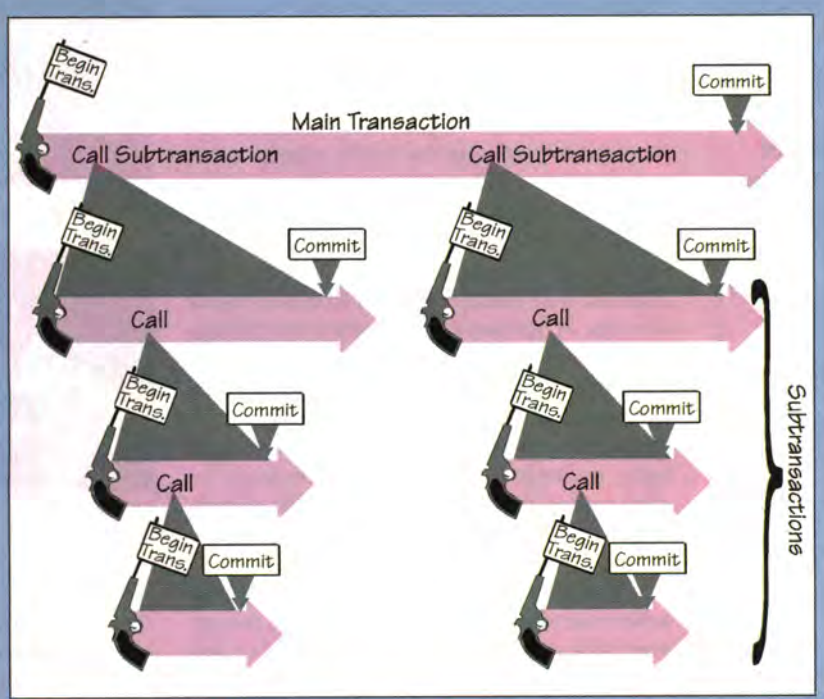


Figure 2. *Nested transactions: one top-level transaction and many dependent subtransactions.*

- *A transactional server* is a collection of one or more objects whose behavior is affected by the transaction but have no recoverable states or resources of their own. The ORB implicitly propagates the transaction's context whenever these objects call a recoverable resource. A transactional server does not participate in the completion of the transaction, but it can force the transaction to be rolled back.
- *A recoverable server* is a collection of one or more objects whose data (or state) is affected by committing or rolling back a transaction. Recoverable objects are transactional objects with resources to protect. Examples of recoverable resources are transactional files, queues, or databases. Recoverable objects use the register method invocation to tell the transaction service that a recoverable resource has just joined the transaction whose context was propagated in the client call. In addition, recoverable objects provide methods that are used by a transaction coordinator (the coordinator is the transaction service) to orchestrate an ORB-mediated, two-phase commit protocol.

OTS is seamlessly integrated with the ORB mechanisms. It relies on the ORB to automatically propagate the transaction context. Notice that the scope of a transaction is defined by a transaction context that is shared by the participant objects. The transaction context is a pseudo-object that's maintained by the ORB for each ORB-aware thread. The context is null when there is no transaction associated with a thread. The transaction service manages and propagates the transaction context with help from the ORB.

OTS provides "IDL-ized" interfaces for the objects that make up the transaction service. So it is possible for clients and transactional objects to get more intimately involved in the details of the transaction propagation via explicit method invocations. However, most transactions will depend on the ORB to transparently do all the work using its built-in facilities. Fewer interventions means better performance.

THE OTS INTERFACES

Figure 4 shows the four key interfaces of OTS; we left out the minor ones. Here is a brief description of what each of these interfaces do:

- **Current** defines a CORBA pseudo-object, which makes it easy for clients to use OTS. Clients invoke `begin` and `commit` to start and end a transaction. The ORB will transparently propagate the context of the pseudo-object to the transaction service and to all participants in the transaction. The context contains an ID that uniquely identifies the transaction; it also contains status information. The client can invoke `rollback` to abort the transaction. It can `suspend` the transaction to stop propagating the context with each message; it can `resume` it when it wants the context to be propagated. `Get_control` returns a **Control** object that you can use to directly interact with the transaction service—a recoverable server typically invokes this method to obtain a reference to its transaction coordinator. A top-level transaction invokes `set_timeout` to define maximum elapsed time (in seconds) for its subtransactions to complete before it aborts them.
- **Coordinator** is implemented by the transaction service. Recoverable objects use it to coordinate their participation in a transaction with the OTS. A server invokes `register_resource` to participate in a transaction. If it supports nested transactions, it instead invokes `register_subtran_aware`. It invokes `create_subtransaction` to create a nested transaction that's a child of the current transaction. It invokes `rollback_only` to abort the entire transaction. The `get_` and `is_` operations are useful to servers that need to explicitly control their participation in a transaction. The `hash_` operations return a handle to the current transaction. Before invoking `register_resource`, servers use the hashed value to find out if the current resource is already registered.
- **Resource** is implemented by a recoverable server object to participate in a two-phase commit protocol. OTS uses the two-phase commit protocol to coordinate a transaction's commit or abort across multiple server objects so that they either all fail or all succeed. To do this, OTS centralizes the decision to commit but gives each participant the right of veto. It's like a Christian wedding: You're given one last chance to back out of the transaction when you're at the altar. If none of the parties object, the marriage takes place.

In the first phase of a commit, OTS invokes `prepare` on all the partic-

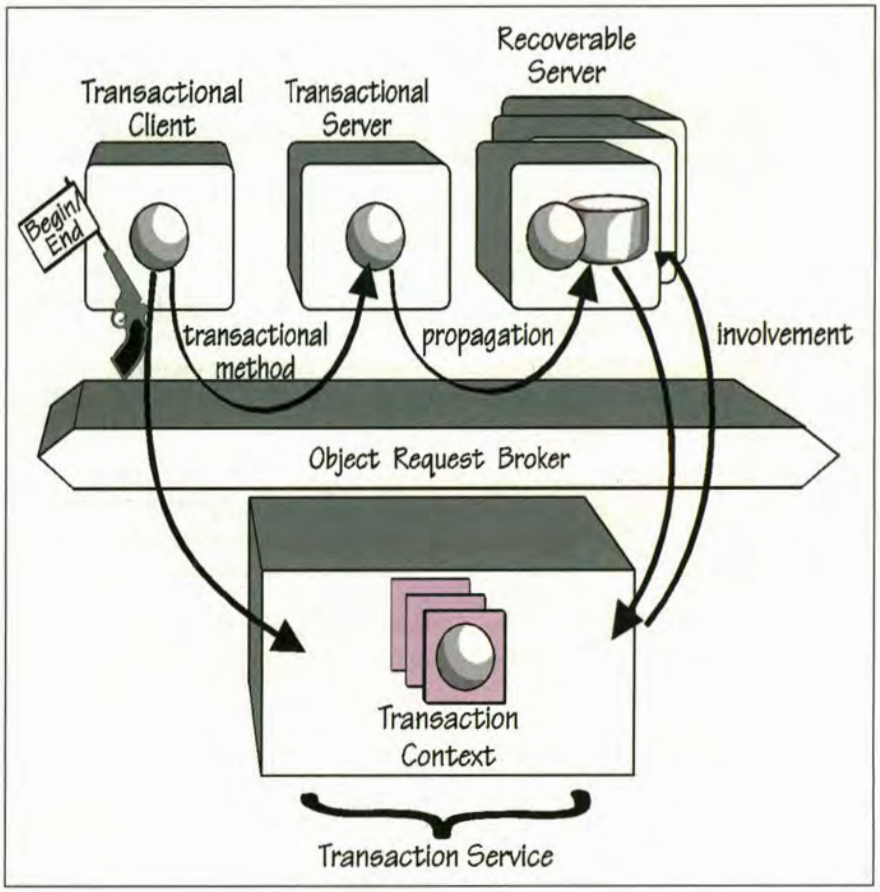


Figure 3. *Object transaction service: meet the players.*

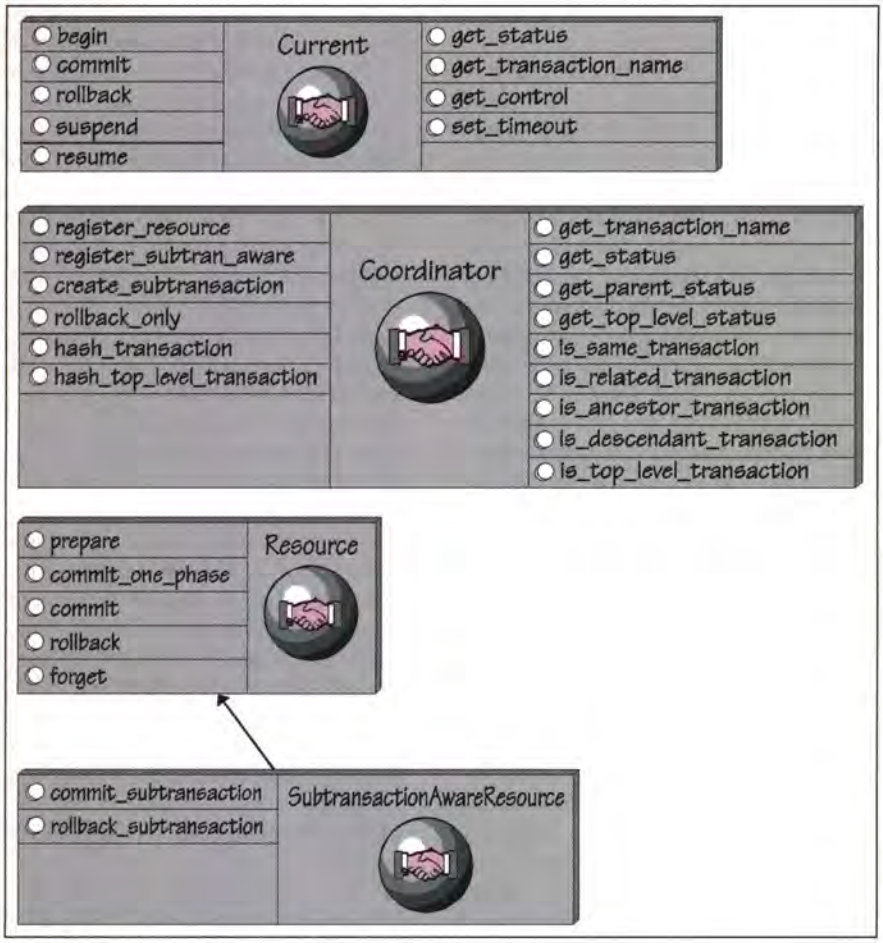


Figure 4. *Object transaction service: the key interfaces.*

ipant resource objects. Each resource object returns in a parameter a "vote" value that is either `vote_commit`, `vote_rollback`, or `vote_readonly`. Based on the vote's outcome (everyone has veto power), the transaction service either issues `commit` or `rollback`. It can also issue a `forget` to a resource object that's fuzzy about its outcome. If the coordinator has only a single registered resource, it avoids the two-phase commit altogether and invokes `commit_one_phase` instead.

- **SubtransactionAwareResource** is implemented by a recoverable server object with nested transaction behavior. This interface is derived from the **Resource** interface. It adds two new methods: `commit_subtransaction` and `rollback_subtransaction`. These methods are invoked when subtransactions are complete. Subtransaction aware server objects must first register with the **Coordinator** object by invoking `register_subtran_aware`.

You should also be aware of the existence of a **TransactionalObject** interface. This is an abstract class that defines no operations. What is it good for? It's actually a very important class. It serves as a marker, which objects use to indicate that they're transactional. To make your object transactional, you simply inherit from the **TransactionalObject** class. The ORB will then propagate the transaction context associated with a client's thread whenever the client invokes any method on your object. Note that the ORB passes this context in a special field (the context field) that is totally transparent as far you're concerned. It's something ORBs do implicitly.

AN OBJECT TRANSACTION SCENARIO

Are you getting overwhelmed with all these interfaces? We will explain how these interfaces are really used with an annotated scenario (too bad we can't animate an article). Figure 5 shows a scenario of a client doing a debit against one server object and a credit against another one. The example is "get $100,000 dollars out of Jeri's account in Bank A and put it in your account in Bank B." You don't want this to fail, do you? The objects representing Bank A and Bank B are recoverable server objects that multiply inherit from the **Resource** class and the **TransactionalObject** abstract class (remember, this is the indicator that does nothing). We also show two transaction service objects that are instances of the **Current** and **Coordinator** classes. The client initiates this transaction across the two servers and issues a `commit` when it's done. Hopefully, the money will get tucked away safely in your account. Here's the explanation that goes along with the numbers in the picture:

1. Client begins transaction. The client invokes `begin` on the **Current** pseudo-object. The ORB passes this information to the transaction service that maintains a **Current** object for each active transaction.

2. Client makes a `debit` on Bank A's object. The client invokes a debit method on an object in Bank A. This object implements the logic of the transaction (something you write), but it also inherits its behavior from the OTS classes **Resource** and **TransactionalObject**. So we're dealing with a recoverable resource object.

3. Bank A's recoverable object registers its resource. The recoverable server object first invokes the method `get_control` on the **Current** object (the shorthand notation is `Current::get_control`) to obtain a reference to the **Coordinator** object. Then it invokes `Coordinator::register_resource` to register with the coordinator object for this transaction. The coordinator keeps track of all the participants.

4. Client makes a credit on Bank B's object. The client invokes a `credit` method on an object in Bank B. This object implements the logic of the transaction (again, something you write) but also inherits its behavior from the OTS classes **Resource** and **TransactionalObject**. So we're dealing with a recoverable resource object.

5. Bank B's recoverable object registers its resource. It's a repeat of what Bank A's server just did.

6. Client issues a `commit`. The client invokes the method `Current::commit`. The ORB informs the transaction service that the transaction has ended.

7. **Coordinator** performs phase 1 of two-phase commit. The coordinator invokes each participant's `prepare` method to get a vote from all the participants on whether this transaction should be committed for posterity (that is, you get your $100,000). Let's give this scenario a happy ending by assuming everyone returns a `"vote_commit."`

8. Coordinator performs phase 2

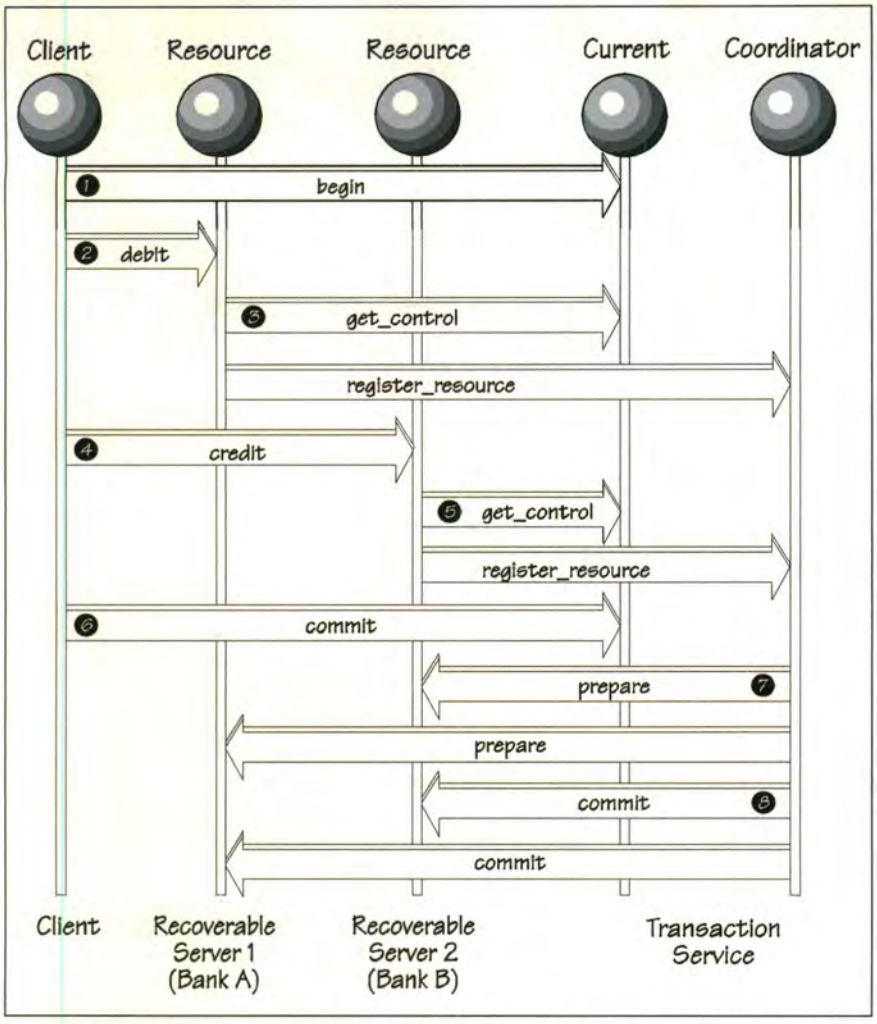


Figure 5. *Object transaction service: a two-phase* `Commit` *scenario.*

of two-phase commit. The coordinator tells all the participants to commit. You now have $100,000 more to spend in your bank account.

Are you all feeling richer, or just tired? It would have been better for Jeri if the coordinator had issued a roll-back instead of a commit. She could have kept her $100,000. Luckily, our scenarios are only fiction. This happy ending concludes our OTS story.

CONCLUSION

In addition to being important at the system level, OTS will redefine the way that we build our client/server middleware. For starters, it will encourage developers to use transactions pervasively. Transactions are now part of the ORB. As a result, most objects that live on the ORB will be transactional. The ubiquitous use of transactions on the ORB will result in TP Monitors morphing into ORBs (or vice versa). This means that ORBs may very well be the next generation TP Monitor.

Jeri Edwards, Dan Harkey, and Bob Orfali are authors of the Essential Client/Server Survival Guide *(Wiley, 1994)—winner of the Jolt Cola Award for best book of 1994. Harkey and Orfali also wrote* Client/Server Survival Guide with OS/2 *(Wiley, 1994) and* Client/Server Programming with OS/2 *(Wiley, 1993). Their latest book is* The Essential Distributed Objects Survival Guide *(Wiley, 1996). Both have been developing client/server applications and tools for the last nine years and are currently working on the application of distributed object technology. They are affiliated with IBM Austin, Tex. Edwards is the director of transaction processing and client/server software development at Tandem Computers, Cupertino, Calif.*

REFERENCES

Orfali, Robert, Dan Harkey, and Jeri Edwards. *Essential Distributed Objects Survival Guide,* Wiley, 1996, (IBM publication number SR28-5898).

Orfali, Robert and Dan Harkey. "Client/Server with Distributed Objects," *BYTE.* April, 1995.

Cobb, Ed. "Objects and Transactions: Together at Last," *Object Magazine.* January, 1995.

Sessions, Roger. *Object Persistence,* Prentice Hall, 1996.

If you would like more information on transactions, we recommend Jim Gray's and Andreas Reuter's *Transaction Processing Concepts and Techniques* (Morgan Kaufmann, 1993). For a lighter introduction, try our *Essential Client/Server Survival Guide,* Wiley, 1994, (IBM publication number SR28-5572).

Searching for the Cure.

Cancer sounds like such a grown-up disease, but each year, more than 6,000 American children will be stricken.

The doctors and scientists at St. Jude Children's Research Hospital are working to wipe out cancer and other catastrophic childhood diseases forever. In fact, research and treatments developed at St. Jude Hospital have already made childhood cancer a survivable disease for thousands of children.

But until *every* child can be saved, the battle against cancer must continue.

To learn more about the life-saving work of St. Jude Hospital, please call **1-800-877-5833.**

ST. JUDE CHILDREN'S RESEARCH HOSPITAL
Danny Thomas, Founder

Configuring DB2/2 for Novell's NetBIOS.

By JEFFREY ALTMAN

DB2/2 Communication Using Novell's NETBIOS Over IPX

Last year, while working on the NETBIOS interfaces in C-Kermit for OS/2 communications software, I came across a friend who was having trouble installing an IBM Database2 OS/2 1.2 (DB2/2) server on a Novell Netware 3.11 wide-area network of OS/2 and DOS workstations. DB2/2 uses the NETBIOS Network application programming interface (API) to enable its servers to communicate with its clients on the DOS/Windows and OS/2 platforms. The problem was that clients on different subnets could not communicate with the server.

The OS/2 workstations were using IBM Multi-Protocol Transport Services (MPTS) and Novell Netware Requester 2.11 to provide network access. The DOS workstations were using a combination of Novell's Netware Client Kit for DOS/Windows and IBM Network Transport Services for DOS (NTS/DOS). All of the systems were using IBM's implementation of NETBIOS for database network communication.

In the years since its introduction in 1984, NETBIOS has become a portable networking interface that is independent of the underlying networking protocols (session and link layers). NETBIOS has been implemented on top of several protocols including, NETBEUI (the original protocol used in the IBM PC Network of the early '80s), IPX (the protocol used in Novell NetWare Networks), and IP (the protocol used on the Internet).

The standard IBM implementation of NETBIOS uses the NETBEUI protocol. While this protocol is fast and easy to use, it does have several downsides. First, installing a NETBEUI NETBIOS protocol on a DOS Netware client requires the purchase of additional software—NTS/DOS. Second, NETBIOS is implemented using a broadcast scheme, which requires that all packets be sent to all workstations. NETBEUI packets are not routable and, therefore, NETBEUI NETBIOS networks are difficult and expensive to configure on networks consisting of multiple subnets.

Users of Novell Netware LANs have a wonderful alternative. They use NETBIOS over IPX instead of NETBIOS over NETBEUI. IPX NETBIOS is included free of charge with the Netware Client Kits for both DOS/Windows and OS/2. By configuring DB2/2 to use IPX NETBIOS instead of NETBEUI NETBIOS, installations with a large number of DOS/Windows clients on a multiple subnet Netware LAN can avoid the expense of purchasing IBM Network Transport Services for DOS (NTS/DOS) and conserve DOS real-mode memory.

In addition, IPX packets are routable, thereby avoiding the requirement that NETBIOS Name Tables be maintained on each LAN segment by the routing equipment in order for NETBIOS Name collisions not to occur. This maintenance of synchronizing multiple name tables would require more expensive routing equipment and increased LAN traffic.

On DOS/Windows clients, all NETBIOS implementations regardless of vendor and underlying protocol use the Int 5Ch and Int 2Ah software interrupts to interface between the application and the NETBIOS driver. This enables any application to use any vendor's NETBIOS. Substituting one NETBIOS implementation for another is therefore a trivial exercise.

In OS/2, NETBIOS is implemented via an API stored in a DLL. Unlike DOS, OS/2 currently has two different standard APIs for implementing NETBIOS: "NETBIOS

Submit" and "NETBIOS 3.0". This results in not every application being able to interface with every implementation of NETBIOS.

DB2/2 was written to use the NETBIOS 3.0 API provided in the IBM MPTS package. Even though Novell provides IPX NETBIOS, it can't be used directly by DB2/2 because it implements the NETBIOS Submit API. However, by installing both MPTS and NetWare Requester for OS/2 (NWR/2) on the workstation in a multiple-stack configuration, it is possible for DB2/2 to use the IPX NETBIOS services.

THE NETBIOS 3.0 API STACK

Under a normal installation of IBM MPTS, three NETBIOS-related files are installed: ACSNETB.DLL, NETBIOS.OS2, and NETBEUI.OS2. The NETBIOS 3.0 API is implemented in ACSNETB.DLL. All programs that are developed to use the NB30 API must load this DLL at run time. NETBIOS.OS2 provides a protocol-independent NETBIOS API called LM10. NETBEUI.OS2 implements the Netbeui protocol, which is called by NETBIOS.OS2. The calling sequence is shown in Listing 1.

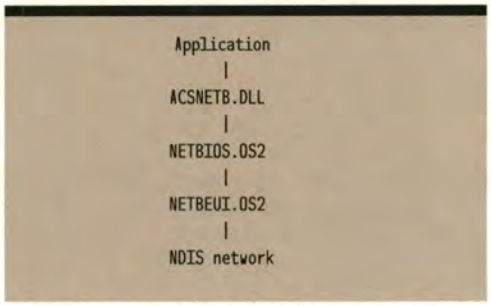
NOVELL'S NETBIOS SUBMIT API STACK

The NETBIOS Submit API was originally developed as part of the IBM OS/2 Extended Edition and then IBM OS/2 Extended Services. NETBIOS Submit was just one of many communications APIs that were included in the original NETAPI.DLL. Novell adopted the NETBIOS Submit API and implemented its own version of NETAPI.DLL with only NETBIOS support. Novell's NETAPI.DLL depends heavily on another DLL, NETSUB.DLL, which actually implements the NETBIOS Submit calls. NETAPI.DLL is just a compatibility layer. NETSUB.DLL, in turn, communicates with Novell's NETBIOS.SYS. NETBIOS.SYS implements a NETBIOS protocol using the Novell IPX protocol. The calling sequence is shown in Listing 2.

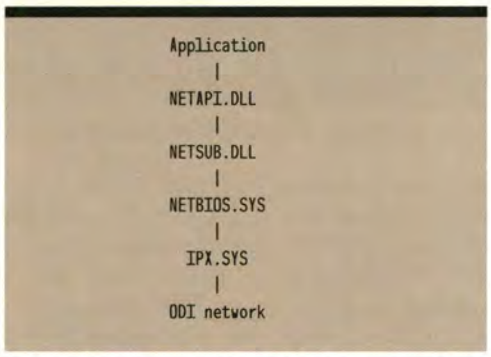
IMPLEMENTING MULTIPLE STACKS WITH LM10 NETBIOS

IBM's LM10 NETBIOS specification (NETBIOS.OS2) is protocol-independent and can implement NETBIOS interface across any protocol compatible with it. Novell's NETSUB.DLL is programmed to look for NETBIOS.OS2 prior to looking for NETBIOS.SYS. Therefore, when both products are installed on the same machine, calls to either NETBIOS 3.0 or NETBIOS Submit interfaces would by default both share the IBM NETBEUI protocol.

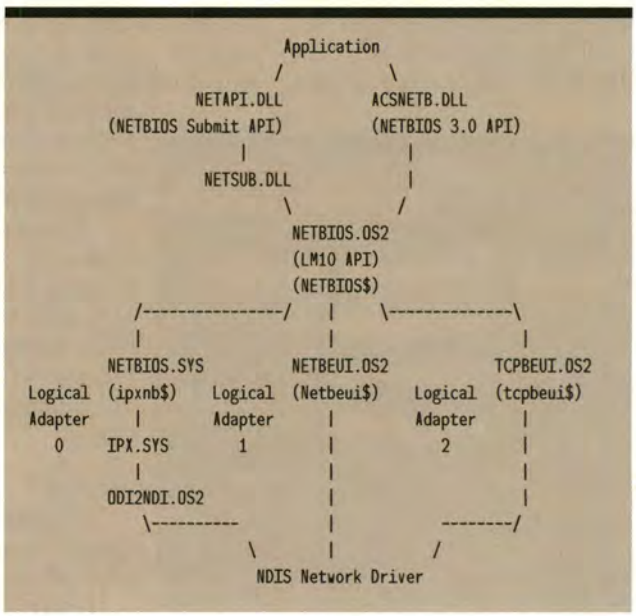
NETBIOS.OS2 can be configured to rec-



Listing 1. *The IBM MPTS NETBIOS protocol stack.*



Listing 2. *The Novell Netware NETBIOS protocol stack.*



Listing 3. *Multiple stack using IBM MPTS and Novell NETBIOS.*

Example 1: Both NETBEUI and IPX NETBIOS implementations and one physical adapter. IPX NETBIOS is the default.

```
[NETBIOS]
   DriverName=NETBIOS$
   Adapter0=ipxnb$,0
   Adapter1=netbeui$,0
```

Example 2: Both NETBEUI and IPX NETBIOS implementations and one physical adapter. NETBEUI NETBIOS is the default.

```
[NETBIOS]
   DriverName=NETBIOS$
   Adapter0=netbeui$,0
   Adapter1=ipxnb$,0
```

Example 3: Only IPX NETBIOS implementation and one physical adapter.

```
[NETBIOS]
   DriverName=NETBIOS$
   Adapter0=ipxnb$,0
```

Example 4: Both NETBEUI and IPX NETBIOS implementations and two physical adapters. NETBEUI NETBIOS is the default. Because IPX NETBIOS retrieves its configuration information from NET.CFG, instead of PROTOCOL.INI, it is not possible to specify a physical adapter binding other than 0.

```
[NETBIOS]
   DriverName=NETBIOS$
   Adapter0=netbeui$,0
   Adapter1=ipxnb$,0
   Adapter2=netbeui$,1
```

Listing 4. *Four examples of using more than one protocol simultaneously.*

ognize more than one protocol. This allows each application to choose the appropriate protocol at run-time regardless of which NETBIOS API it was written to use. This is known as implementing a multiple stack. After proper configuration, the multiple stack could look like Listing 3.

NETBIOS.OS2 retrieves its configuration information from the PROTOCOL.INI file, which is stored in the IBMCOM directory created during the MPTS installation. PROTOCOL.INI consists of sections identified by driver names in square brackets followed by configuration item-and-value pairs indented from the left margin. Most of the settings in PROTOCOL.INI are configured by using MPTS.EXE.

Without a configuration specified in PROTOCOL.INI, NETBIOS.OS2 uses its default configuration to define one Logical NETBIOS Adapter for each Physical Adapter Driver and bind NETBEUI.OS2 to each of the logical adapters.

This default configuration can be overridden by using a text editor, such as E.EXE, to add a [NETBIOS] section to the end of the PROTOCOL.INI file. The format of this new section is:

```
[NETBIOS]
    DriverName=NETBIOS$
<logical adapter>=<driver>, <physical adapter binding>
[,<sessions> [,<commands> [,<names>]]]
...
<logical adapter>=<driver>, physical adapter binding>
[,<sessions> [,<commands> [,<names>]]]
```

where: <logical adapter> is ADAPTER0 through ADAPTER9. The logical adapter numbers must be in sequence.

<driver> is the protocol-specific implementation of NETBIOS. For example, "ipxnb$" for Novell IPX, "netbeui$" for IBM NETBEUI, or "tcpbeui$" for IBM TCP/IP.

<physical adapter binding> is the binding order of the physical adapter driver (that is, IBMTOK_nif NE2000_nif) on the Bindings line in the NETBIOS <driver>`s (NETBEUI_nif or TCPBEUI_nif) section in the PROTOCOL.INI file. This value may range from 0 to 63. However, it is usually 0, since there is rarely more than one physical adapter installed in the machine. When used with the IPX NETBIOS, this value must be 0 because the ipxnb$ driver retrieves its configuration data from the Novell NET.CFG file and not PROTOCOL.INI.

<sessions>, <commands>, and <names> are optional parameters which place an upper limit on the number of session, commands, and names that may be used by the <driver>. These values, if specified, must be greater than or equal to the settings in NET.CFG (Netware NETBIOS section) if <driver> is ipxnb$, or PROTOCOL.INI (Netbeui_nif or Tcpbeui_nif sections) if <driver> is either netbeui$ or tcpbeui$.

By assigning different NETBIOS implementations to different logical adapter numbers, it is possible to use more than one protocol simultaneously. Four examples are shown in Listing 4.

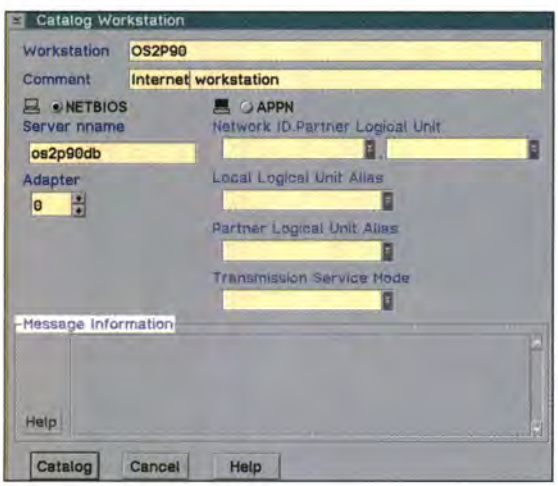


Figure 1. *DB2/2 workstation directory catalog.*

CONFIGURING IBM DB2/2 TO USE MULTIPLE NETBIOS IMPLEMENTATIONS

To enable DB2/2 and the DOS/Windows Client to use multiple logical NETBIOS Adapters, you must add the following line to your CONFIG.SYS and DBDRQLIB\DBDRQLIB.CFG file:

```
SET SQLNADPT=ALL
```

This statement instructs DB/2 to initialize or reset all NETBIOS adapters when it is started. Other valid values are 0, 1, or NONE. NONE tells IBM DB2/2 not to use NETBIOS at all, while 0 or 1 instructs it to use only that particular adapter. The default is 0. DB2/2 can only access adapters 0 and 1. Even though you can define

	Config File	Sessions	Commands	Names
pxnb$	NET.CFG	16	32	24
netbeui$	PROTOCOL.INI	130	254	21
tcpbeui$	PROTOCOL.INI	130	254	21

Table 1. *The fault values of various NETBIOS settings.*

	Config File	Sessions	Commands	Names
ipxnb$	NET.CFG	64	128	128
netbeui$	PROTOCOL.INI	254	255	254
tcpbeui$	PROTOCOL.INI	254	255	254

Table 2. *Maximum values of various NETBIOS settings.*

adapters 2 and 3 to NETBIOS.OS2, DB2/2 can not use them.

To specify the number of client sessions that can be active simultaneously on each NETBIOS adapter add the following line to CONFIG.SYS (DB2/2 Servers only):

```
SET SQLNETB=xx,yy
```

where xx is the number of sessions for Adapter 0 and yy is the number of sessions for Adapter 1. DB2/2 Server has an overhead of six sessions. Therefore, these values must be greater than or equal to six plus the number of clients that will attempt to maintain simultaneous connections to the DB2/2 Server. If xx = 16, then the eleventh client that tries to connect to DB2/2 Server using the driver assigned to Adapter0 will not be able to connect.

CATALOGING DB2/2 WORKSTATIONS ON NETBIOS NETWORKS

NETBIOS workstations are defined within DB2/2 by using the Directory Tool. Open the Workstation Directory and then choose Workstation Catalog (Figure 1). Choose a local name for the remote DB2/2 Server. Select a NETBIOS connection. Provide the NETBIOS name specified when the server was configured. Then, select the logical NETBIOS adapter upon which that server can be found. Remember, if the server is configured to use NETBEUI NETBIOS, a workstation configured to use IPX NETBIOS will not be able to communicate with it.

DB/2 NETBIOS RESOURCE REQUIREMENTS

IBM DB2/2 requires the following resources from each NETBIOS protocol: 4 names, 46 commands, and as many sessions as are specified in the SQLNETB line in the CONFIG.SYS file. If these resources are not available when IBM DB2/2 is started, NETBIOS connections will not be available for that adapter.

The number of commands, sessions, and names must be configured separately for each NETBIOS protocol stack. NETBEUI NETBIOS is configured using MPTS.EXE. IPX NETBIOS is configured by editing the NET.CFG file. (See appropriate documentation associated with each product for further details.)

The default and maximum settings for various NETBIOS implementations are illustrated in Tables 1 and 2. Because Novell NETBIOS defaults to 32 commands, a "Netware NETBIOS" section must be added to the NET.CFG file. In that section, a "commands" statement should be placed, which increases the number of commands to at least 46.

NETBEUI NETBIOS should enable DB2/2 to support a greater number of sessions than IPX NETBIOS. But the maximum values, shown in Table 2, are pipe dreams. Due to memory constraints within the NETBIOS driver, it is unlikely that the maximum values could ever be used.

Listings 5, 6, and 7 provide excerpts from the configuration files of a system that has implemented the techniques previously described.

CONCLUSION

Having successfully configured our NETBIOS implementation for use with IBM DB2/2, we have expanded the range of use of our client server

Continued on page 45

Glossary

IP. Networking protocol used for communication on the Internet.

IPX. Networking protocol used for communication between Novell Netware clients and servers.

LM10. A protocol-independent NETBIOS programming interface at the device driver level.

Logical adapter. The NETBIOS API assumed that only one protocol could exist on a given network adapter card. Logical adapters are used to allow NETBIOS to distinguish between between unique combinations of protocols and physical adapters.

NETBEUI. Networking protocol developed for single subnets LANs in the early '80s.

NETBIOS. A protocol-independent API. Available on all types of networks and operating systems, NETBIOS enables software developers to create client/server applications without being concerned about the type of network the application is using for communication.

Physical adapter binding. A protocol driver may be bound to more than one network adapter card driver. The physical adapter binding value specifies which network adapter is being referred to in the binding list.

OS/2® and AIX Tools at Your Command

The best way to find out what's available for IBM system software platforms.

http://www.mfi.com/softwareguide

Stay on top of it by browsing this software and services directory of products for OS/2, AIX and other IBM supported operating systems.

Sponsored by IBM®, *OS/2 Developer*, *OS/2 Magazine*, and *Software Development*.

OS/2® is a registered trademark of International Business Machines Corporation and is used by Miller Freeman, Inc. under license.

Find It on the Web!


```
SET SQLNETB=16
SET SQLNADPT=ALL
LIBPATH=...D:\MPTN\DLL;D:\IBMCOM\DLL;...
     D:\NETWARE.211;L:\OS2;P:\OS2;
SET PATH=...D:\MPTN\BIN;D:\IBMCOM;...
     D:\NETWARE.211;L:\OS2;P:\OS2;
SET DPATH=...D:\IBMCOM;...
    D:\NETWARE.211;D:\NETWARE.211\NLS\ENGLISH;L:\NLS;P:\NLS;...
SET HELP=...D:\NETWARE.21\NLS\ENGLISH;...
DEVICE=D:\IBMCOM\PROTOCOL\LANPDD.OS2
DEVICE=D:\IBMCOM\PROTOCOL\LANVDD.OS2
DEVICE=D:\IBMCOM\PROTMAN.OS2 /I:D:\IBMCOM
DEVICE=D:\IBMCOM\LANMSGDD.OS2 /I:D:\IBMCOM
REM --- NetWare Requester statements BEGIN ---
SET NWLANGUAGE=ENGLISH
DEVICE=D:\NETWARE.211\LSL.SYS
RUN=D:\NETWARE.211\DDAEMON.EXE
REM -- ODI-Driver Files BEGIN --
DEVICE=D:\IBMCOM\PROTOCOL\ODI2NDI.OS2
REM -- ODI-Driver Files END --
DEVICE=D:\NETWARE.211\IPX.SYS
DEVICE=D:\NETWARE.211\SPX.SYS
RUN=D:\NETWARE.211\SPDAEMON.EXE
DEVICE=D:\NETWARE.211\NMPIPE.SYS
DEVICE=D:\NETWARE.211\NPSERVER.SYS
RUN=D:\NETWARE.211\NPDAEMON.EXE NPSERVERNAME
DEVICE=D:\NETWARE.211\NWREQ.SYS
IFS=D:\NETWARE.211\NWIFS.IFS
RUN=D:\NETWARE.211\NWDAEMON.EXE
DEVICE=D:\NETWARE.211\NETBIOS.SYS
RUN=D:\NETWARE.211\NBDAEMON.EXE
DEVICE=D:\OS2\MDOS\LPTDD.SYS
REM --- NetWare Requester statements END ---
RUN=D:\IBMCOM\LANMSGEX.EXE
DEVICE=D:\IBMCOM\PROTOCOL\LANDD.OS2
DEVICE=D:\IBMCOM\PROTOCOL\LANDLLDD.OS2
TRACEBUF=16
DEVICE=D:\IBMCOM\MACS\IBMTOK.OS2
RUN=D:\IBMCOM\PROTOCOL\LANDLL.EXE
DEVICE=D:\IBMCOM\PROTOCOL\NETBEUI.OS2
DEVICE=D:\IBMCOM\PROTOCOL\NETBIOS.OS2
RUN=D:\IBMCOM\PROTOCOL\NETBIND.EXE
```

Listing 5. *Excerpts from sample CONFIG.SYS file after installation of Novell Netware Requester 2.11 and IBM MPTS.*

```
[NETBEUI_nif]

   DriverName = netbeui$
   Bindings = IBMTOK_nif
   ETHERAND_TYPE = "I"
   USEADDRREV = "YES"
   OS2TRACEMASK = 0x0
   SESSIONS = 48
   NCBS = 190
   NAMES = 64
   SELECTORS = 16
   USEMAXDATAGRAM = "YES"
   ADAPTRATE = 1000
   WINDOWERRORS = 0
   MAXDATARCV = 16384
   TI = 30000
   T1 = 500
   T2 = 200
   MAXIN = 32
   MAXOUT = 64
   NETBIOSTIMEOUT = 500
   NETBIOSRETRIES = 8
   NAMECACHE = 16
   PIGGYBACKACKS = 1
   DATAGRAMPACKETS = 2
   PACKETS = 350
   LOOPPACKETS = 1
   PIPELINE = 5
   MAXTRANSMITS = 6
   MINTRANSMITS = 2
   DLCRETRIES = 5
   FCPRIORITY = 5
   NETFLAGS = 0x0

[NETBIOS]

   DriverName = NETBIOS$
   ADAPTER0 = netbeui$,0
   ADAPTER1 = ipxnb$,0
```

Listing 6. *Sample [NETBIOS] and [NETBEUI_nif] sections from PROTOCOL.INI file.*

database applications without increasing the burden on the network administrators or bearing additional software expense.

```
NetWare NetBIOS
  internet on      ; allows routing of IPX NETBIOS packets
  sessions 64      ; maximum number of sessions increased to 64
  commands 128     ; max commands increased to 128
  names    128     ; max names in name table increased to 128
```

Listing 7. *Sample "Netware NETBIOS" section from NET.CFG file.*

REFERENCES

Client Server Programming with IBM OS/2 2.1 3rd Edition, Orfali and Harkey, VNR.

Novell Netware Requester for OS/2 Installation Guide, (OS2BOOK.INF) [part of Netware Requester for OS/2].

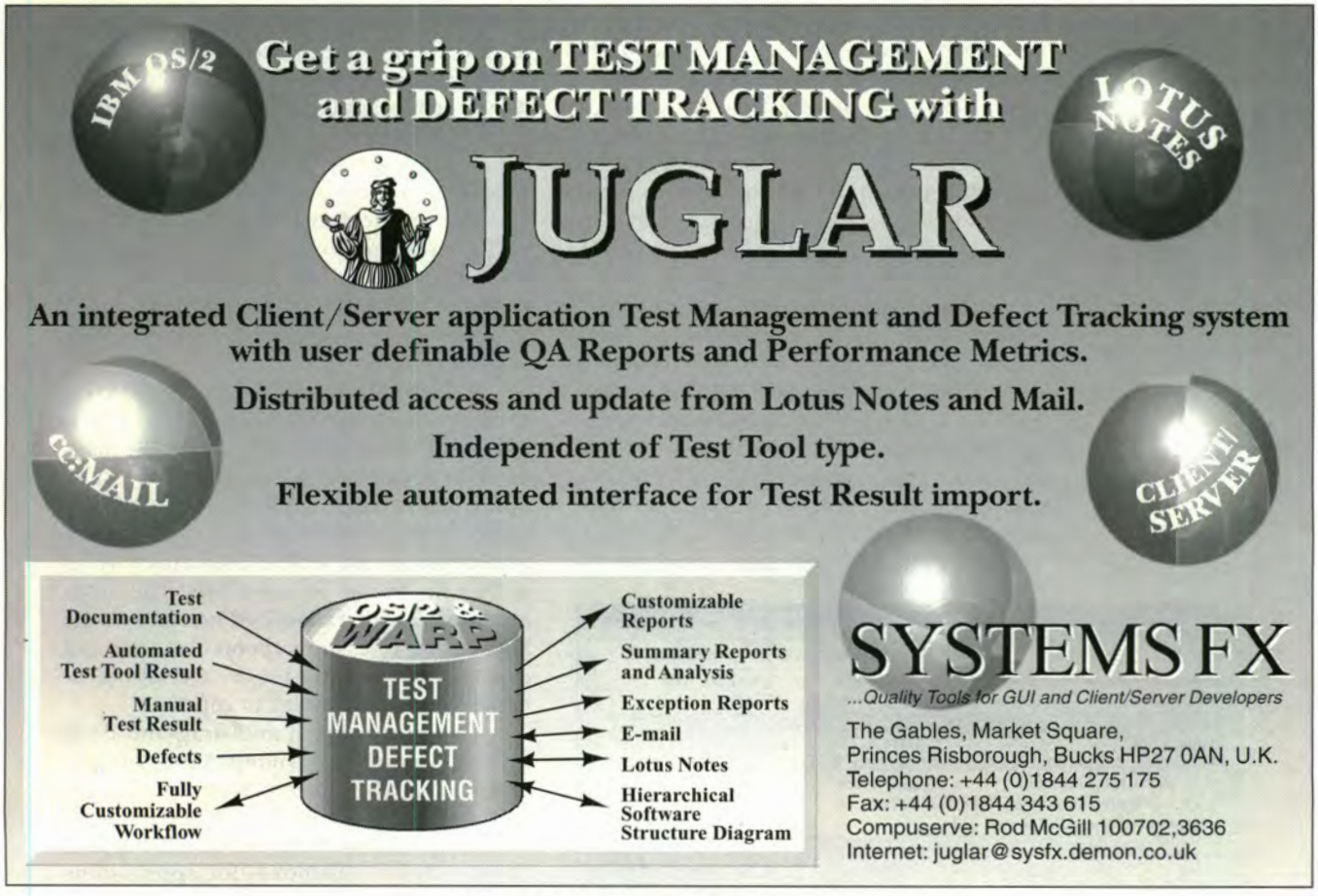
IBM MPTS Configuration Guide, (A3V10M02.INF) [part of MPTS].

IBM Database2 OS/2 Guide, [ships with DB2/2].

MPTS is shipped as part of the following IBM packages: IBM LAN Manager 4.0 and IBM OS/2 WARP Connect. MPTS provides a NETBEUI NETBIOS implementation. OS/2 WARP Connect includes NETBEUI NETBIOS, IPX NETBIOS, and TCP/IP NETBIOS.

It is important to note that the techniques used in this article to enable DB2/2 to use IPX NETBIOS are not limited to Novell's implementation. They can also be used to allow DB2/2 to use IBM NETBIOS for TCP/IP or any other NETBIOS implementation consistent with the LM10 specification. Nor is DB2/2 the only application that can take advantage of multiple NETBIOS stacks. Any NETBIOS-aware application, which allows the specification of a NETBIOS Adapter, can take advantage of these benefits. This includes DOS and Windows applications running under OS/2.

Jeffrey Altman *is currently a senior software designer at Columbia University's Kermit Software Development Group. He has been developing cross-platform communication software on IBM OS/2 and other PC operating systems for several years. He is currently pursuing a Ph.D. in Computer Science at Columbia University. His research topics include object-oriented techniques for distributing copyrighted or licensed information across heterogeneous networks of computers. He can be reached at (212) 854-1344 or via Internet at jaltman@columbia.edu.*



Circle Reader Service Number 12

Some useful programming tips: implementing multiple columns and drag/drop features in a PM listbox.

By MARK MCMILLAN

A Multicolumn Drag/Drop Listbox

In the quest for ever easier user interfaces, the listbox is a natural target for improvement in OS/2 PM. Mark Benge and Matt Smith described many listbox improvements in a series of articles starting in the January/February 1994 issues of *OS/2 Developer*. Some of those improvements, most notably the removal of the 64-K data limit, were implemented in OS/2 Warp 3.0.

Two particular concepts that they described, however, were never actually incorporated in their listbox replacement. (Hey, they are busy guys!) In this article, I will describe a custom PM control, which implements a true multicolumn listbox, and a subclass procedure, which performs drag-and-drop listbox reordering. These items can be used separately or together to create a multicolumn, drag-and-drop listbox.

Unlike the Benge/Smith approach of writing a control from scratch, my techniques make extensive use of the existing PM listbox. The original concept of the MCLB was created by Charles Cooper of the IBM Warwick Development Group (IBM U.K. Ltd.). The drag-and-drop concepts were created by Allan Warren of IBM Yorktown Research. Working at IBM in Research Triangle Park, I rewrote both implementations, added some features, and simplified the programming interfaces.

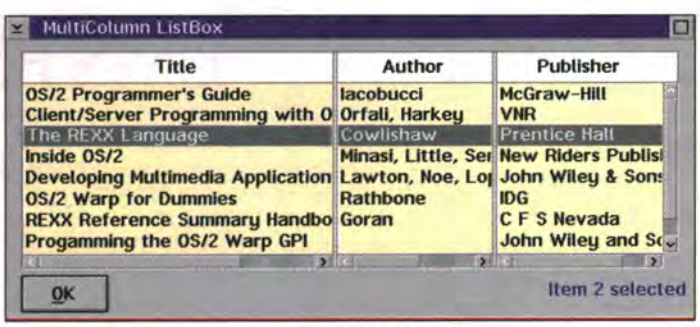


Figure 1. *A multicolumn listbox with sizeable columns.*

SEEKING ROWS AND COLUMNS

Applications often need to display data in a column format selectable by rows. Developers that prefer to avoid the complexity of the container control have tried various means to achieve a multicolumn effect with the listbox control. Common methods include using a fixed-pitch font so that characters align into columns or using an owner-drawn listbox with the application rendering the text of each item. Both of these methods have serious drawbacks in terms of visual presentation or code complexity.

The multicolumn listbox control presented here preserves the simplicity of the PM listbox but adds a visually pleasing multicolumn capability, including proportional font support, movable column dividers, and column titles.

A second common need for the listbox is to display a list of information in a specific order and allow the user to reorder the items or move them from one list to another. Many applications faced with this requirement resort to an elaborate set of pushbuttons or other controls to let the user move items within the list or between lists. The drag-and-drop function described here allows the application to easily support the drag and drop of items within a list (for reordering) or between lists (for moving or copying).

The multicolumn and drag-and-drop listbox supports are independent of each other and can be used separately, or they can be combined to create a multicolumn drag-and-drop listbox. Both functions support "owner-drawn" listboxes for applications that need nontext listbox items such as bitmaps or other graphics.

MULTICOLUMN LISTBOX OVERVIEW

A MultiColumn ListBox (MCLB) is a composite control consisting of a variable number of listboxes, column separators, and column headings (Figure 1). The overall control organizes these components into a single user interface component. The control looks more like a container than a listbox from the user's point of view. However, from a programmer's point of view, the MCLB is much more like a listbox (and, in fact, uses the usual listbox messages). The user operates an MCLB much like a details-view container. Records can be selected, and the splitbars moved to change the relative column sizes. When the user clicks on an item in any column of the MCLB, the entire record (row) is selected. A single vertical scroll bar on the right edge scrolls the records up and down. The MCLB can optionally have horizontal scroll bars at the base of each column.

An MCLB comes in two basic styles: one with columns, which are sizable via splitbars, and one with fixed-width columns and no split bars. Figure 2 shows an MCLB using fixed-width columns. Because the MCLB is based on the real PM listbox control, any of the usual listbox styles can also be used with an MCLB, such as `LS_HORZSCROLL` to place horizontal scroll bars at the bottom of each column or `LS_MULTIPLESEL` to allow multiple record selection. `LS_OWNERDRAW` style is also supported if the application wants to draw non-text items into one or more columns, such as the bitmaps shown in Figure 3. The MCLB supports any number of columns, but performance decreases in a linear fashion as the number of columns increase.

Columns can be resized by positioning the pointer over a splitbar and dragging it to a new position. The columns adjacent to the splitbar are then resized (other columns are unaffected). The Alt-arrow keys can be used to move the splitbar with the Ctrl-arrow keys moving from one splitbar to the next.

Each column of an MCLB shrinks or grows in proportion to its relative size when the MCLB window is resized. For example, if a column was 20% of the width of the MCLB before resizing the window, its width would be automatically adjusted to remain 20% of the window size. This resizing algorithm can be modified through the use of style flags.

The MCLB supports the use of the system font and color palettes to allow drag and drop of fonts and colors. An MCLB can use different fonts and colors for the column headers and the data records. All columns are updated to use the same font or color when a font or color is dropped on an MCLB column.

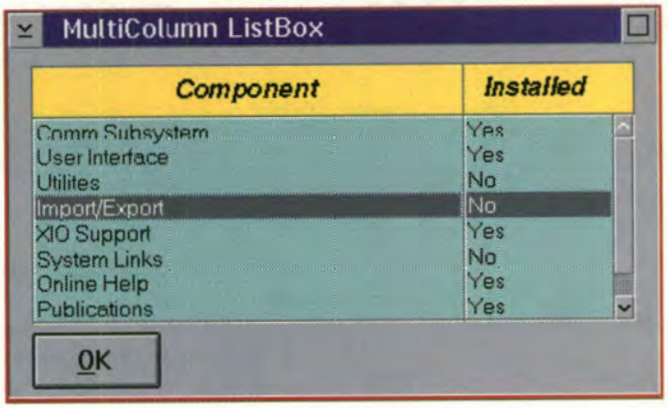


Figure 2. *An MCLB with fixed columns.*

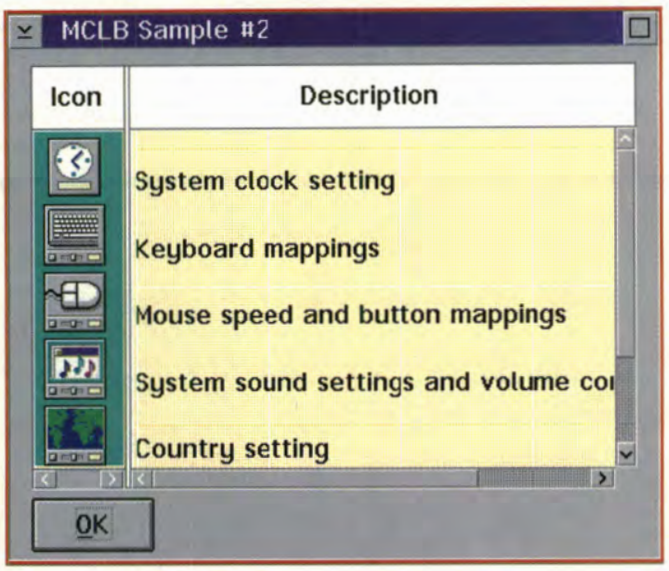


Figure 3. *An MCLB with bitmaps.*

```
typedef struct _MCLBINFO {
  char   *Titles;                  // Title strings (TabChar separated)
  char   TitleFont[MAX_FONTLEN];   // Title font (null for default)
  char   ListFont[MAX_FONTLEN];    // List  font (null for default)
  ULONG  TitleBColor;              // Title background color
  ULONG  TitleFColor;              // Title foreground color
  ULONG  ListBColor;               // List  background color
  ULONG  ListFColor;               // List  foreground color
  LONG   *InitSizes;               // Ptr to array of initial sizes
  char   _Reserved[64];            // Reserved for future use
  USHORT Cols;                     // Number of columns
  char   TabChar;                  // Data column separator character
  char   _Padd;                    // Padd for separator character
} MCLBINFO;
```

Listing 1. *MCLB initialization structure.*

The application can respond to user selections, de-selections, and the ENTER key like a normal PM listbox.

PROGRAMMING AN MCLB

If you are familiar with the PM listbox, then you already know most of what you need to use an MCLB. The MCLB uses the usual listbox messages for inserting text (`LM_INSERTITEM`) and for determining selection status (`LM_QUERYSELECTION`). A few differences exist, however, in creating an MCLB, and some additional control messages are generated.

From a programming viewpoint, the MCLB is logically a single listbox. Each listbox item represents an entire row of the MCLB. When an item is selected in the MCLB, the entire row is highlighted and considered selected. Items in a listbox are addressed by their zero-based index (position in the list). This index corresponds to the zero-based row number in an MCLB. When the application queries the text for item 5, for example, the MCLB returns a single string, which represents all the text in row 5, and each column's text is delimited with a separator.

```
case WM_INITDLG: {
  SWP Pos;                         // Pos of placeholder
  MCLBINFO MCLBInfo;               // MCLB create structure
  LONG     ColSize = {1L, 1L, 2L}; // Last column 2X others

  // Build MCLB create data -- just required fields
  memset(&MCLBInfo, 0x00, sizeof(MCLBINFO));
  MCLBInfo.Cols    = 3;            // Number of columns
  MCLBInfo.TabChar = '/';          // Col separator
  MCLBInfo.Titles  = "First/Mid/Last";// Titles

  // Place holder has same ID as MCLB will have.  Get
  // its position in the dialog, then we are done with it.
  WinQueryWindowPos(WinWindowFromID(hwnd, ID_MCLB), &Pos);
  WinDestroyWindow(WinWindowFromID(hwnd, ID_MCLB));

  // Create MCLB in place of place holder
  MCLBHwnd = MCLBCreateWindow(
                 hwnd,           // Dialog is parent
                 hwnd,           // and owner.
                 WS_VISIBLE |    // Make MCLB visible,
                   WS_TABSTOP |  //   be a dialog tab control,
                   LS_HORZSCROLL //   have horizontal scroll bars,
                   MCLBS_NOCOLRESIZE, // not user-sizable columns.
                 Pos.x, Pos.y,   // Position
                 Pos.cx,Pos.cy,  // Size
                 HWND_TOP,       // Top of siblings
                 ID_MCLB,        // ID of MCLB control
                 &MCLBInfo);     // Create data

   //... remainder of dialog initialization ...//

   return (MRESULT)TRUE;
   }
```

Listing 2. *Creating an MCLB in a dialog.*

CREATING AN MCLB

The application must first create a data structure with the MCLB initialization data, including specifications for the number of columns, column titles, a separator character, and style flags. Listing 1 shows the `MCLBINFO` data structure that the application constructs. The application then calls `MCLBCreateWindow()`, passing the usual window-creating information such as window size, parentage, and the `MCLBINFO` structure. A window handle is returned if the MCLB is successfully created.

Most of the fields of the `MCLBINFO` structure are obvious, but a few benefit from some explanation:

- The **TabChar** field contains a single non-null character that will be used to separate the column data in a single record. As will be discussed later, each record of the MCLB is logically a single string, with this character used to separate the data into columns.
- The `/InitSizes/` field is a pointer to an array of `LONG` values. There must be as many elements of this array as there are columns specified in the `/Cols/` field. These values establish the initial size (width) of the columns and are relative for an `MCLBS_SIZEMETHOD_PROP`-style MCLB, since the columns will be proportionally expanded to fill the width of the control. For example:

```
LONG ColSizes[3] = {2L, 1L, 1L};
```

  will create a listbox in which the first column is twice the size of the second and third columns. The initial size values define the pixel widths of the columns for a style of MCLBS_SIZEMETHOD_LEFT. Any leftover space will be given to the left column.

The MCLB does not currently support direct creation from a dialog template. However, it is very easy to get the same effect by defining a static rectangle (or any other control) in the dialog template to act as a placeholder. Listing 2 shows a `WM_INITDLG` sequence used to create an MCLB and then replace a static dialog control with it.

MCLB ITEM STRINGS

In keeping with the listbox programming model, each item (record) in the

Message	Description
MCLB_SETTITLES	Set new column titles (mp1 is pointer to delimited string).
MCLB_SETTITLEFONT	Set title font (mp1 is pointer to font name/size or NULL to use default font).
MCLB_SETLISTFONT	Set list font (mp1 is pointer to font name/size or NULL to use default font).
MCLB_SETTITLECOLORS	Set title colors (mp1 is foreground RGB, mp2 is background RGB). If mp1==mp2 then colors are reset to defaults.
MCLB_SETLISTCOLORS	Set list colors same format as title colors.
MCLB_SETCOLSIZES	Set column sizes (mp1 is pointer to array of LONG).
MCLB_QUERYCOLSIZES	Query current column sizes (mp1 is pointer to array of LONG). Returned values are pixels.
MCLB_QUERYINFO	Query info (mp1 is pointer to MCLBINFO structure). App must free title string and array of sizes.
MCLB_QUERYSTYLE	Query MCLB style flags.
MCLB_QUERYFULLSIZE	Query width of sum of columns (for example, width available for columns)
MCLB_QUERYCTLCOL	Query source of current WM_CONTROL message (column number). Must be sent, not posted, and is only valid during WM_CONTROL.

Table 1. *MCLB messages.*

MCLB control has a single text string associated with it. The string defines the text that is displayed on that row of the MCLB. A single separator character is used to delimit where the text for each column begins and ends in the string. The separator can be any binary value except zero and is specified in the MCLBINFO structure when the MCLB is created.

Text strings are assumed to contain the separator characters when inserting or setting text in the MCLB. A null (empty) column is denoted by two adjacent separator characters in the string. For example, the following item strings could be used for a 3 column MCLB with a / (slash) for the separator:

```
John//Smith
Steve/A/Campbell
Eric
```

The MCLB always returns fully delimited item strings when text is queried. The previous sample would be returned from an LM_QUERYITEMTEXT message as:

```
Eric//
```

INSERTING RECORDS

Inserting items in an MCLB is done with the standard PM listbox messages. The listbox insert messages are extended with an extra field to specify the column for sorting. The extension is done by using part of the message parameters that are unused by the standard PM listbox messages. Here is an example of a new record inserted into a three-column listbox. The record is inserted so that the third column remains in sort order:

```
WinSendMsg(MCLBHwnd, LM_INSERTITEM,
MPFROM2SHORT(LIT_SORTASENDING, 3),
MPFROMP("Steve/A/Campbell"));
```

This extension is defined for the LM_INSERTITEM and LM_INSERTMULTITEMS messages.

MCLB NOTIFICATIONS

The MCLB control generates all of the usual listbox WM_CONTROL notification messages such as LN_SELECT and LN_ENTER. In addition, the MCLB control generates some new messages to notify the owner of MCLB-specific events:

- MCLBN_COLSIZED is generated when the user changes a column size by using a splitbar.
- MCLBN_CUSTOMSIZE is generated when the MCLB window is resized and the MCLBS_SIZEMETHOD_CUSTOM style is used. The application responds by setting the new sizes for all the columns.
- MCLBN_PPCHANGED is generated when a presentation parameter (fonts or colors) in the MCLB has been changed.

It is sometimes useful to know which column of the MCLB caused the generation of a particular notification message. The application can send the MCLB an MCLB_QUERYCTLCOL message to determine which column caused the WM_CONTROL message. This query must be sent (not posted) while the application is processing the WM_CONTROL message.

MCLB MESSAGES

All the usual listbox messages (LM_*) may be sent to the MCLB. The listbox messages can be used to select items (rows), query the text, query the selected items, change the text of an item, or set/query item handles.

The MCLB supports some additional messages for MCLB-specific functions plus the standard listbox messages. Table 1 lists all these MCLB-specific messages that can be sent to an MCLB. These include messages to set fonts and colors for the titles and data lists, setting new title strings, and querying various information about the size and status of the columns.

OWNER-DRAW MCLB

The MCLB control supports owner-drawing of the data columns. The application is responsible for drawing all the rows and columns of the data items of an MCLB with the LS_OWNERDRAW style. (This style cannot be applied to a single column.) LS_OWNERDRAW can be used when the application wants to draw nontext items in one or more columns of the MCLB.

The owner of an owner-draw MCLB will receive WM_MEASUREITEM and WM_DRAWITEM messages just like a stan-

```
case WM_DRAWITEM:
  /* We get this message when an item in a column needs drawing.   */
  /* For column 1 we paint the supplied presentation space with a  */
  /* background and the bitmap.  For column 2 we just return FALSE */
  /* so the listbox will draw the item text.                       */

  if (SHORT1FROMMP(mp1) == ID_MCLB) {  // Draw is from our MCLB
    OWNERITEM *OwnerInfo;
    POINTL    Point;
    USHORT    Col;

    OwnerInfo = (OWNERITEM *)mp2;     // mp2 is ptr to OWNERITEM

    // Get column this draw is for
    Col = SHORT2FROMMP(mp1);
    if (Col != 1)  // Let listbox draw text in 2nd column
      return (MRESULT)FALSE;

    // If redraw needed, fill background and draw bitmap

    if (OwnerInfo->fsState == OwnerInfo->fsStateOld) {
      WinFillRect(OwnerInfo->hps, &(OwnerInfo->rclItem), CLR_CYAN);

      // Center bitmaps in the item rectangle
      Point.x = (OwnerInfo->rclItem.xRight
                - OwnerInfo->rclItem.xLeft)/2
                - BmpHSize/2 + OwnerInfo->rclItem.xLeft;
      Point.y = (OwnerInfo->rclItem.yTop
                - OwnerInfo->rclItem.yBottom)/2
                - BmpVSize/2 + OwnerInfo->rclItem.yBottom;

      // Bitmap handle is item handle set when item was inserted
      WinDrawBitmap(OwnerInfo->hps, (HBITMAP)OwnerInfo->hItem,
                    NULL, &Point, 0L, 0L, DBM_NORMAL);
    }
    return (MRESULT)TRUE;  // Tell listbox that we drew the item
  }
  break;
```

Listing 3. *Processing* `WM_DRAWITEM` *for an MCLB.*

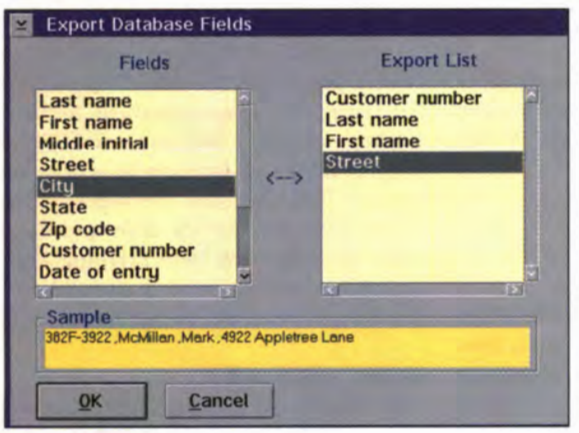


Figure 4. *Use of drag/drop listboxes.*

dard PM owner-draw listbox. These messages will occur for each row and column of the listbox. (This is one place where the single listbox programming model does not apply.) These messages normally carry the item index in the first SHORT of mp1. The message is extended for an MCLB to carry the column number in the second SHORT of mp1. The application can examine both parts of mp1 to determine which row and column is to be drawn.

Listing 3 shows the processing of a `WM_DRAWITEM` message for a two-column MCLB. The MCLB item handles are used to keep a bitmap handle in this sample application (SAMP2 in the toolkit SAMPLES directory). This handle is then used to render the bitmaps for column 1 during `WM_DRAWITEM` processing. For column 2, the sample just returns `FALSE` from `WM_DRAWITEM`, so the MCLB will draw the item text as usual. (The application must still properly handle the `WM_MEASUREITEM` message for column 2 even though it is drawn by the MCLB.)

Gaining access directly to the underlying listbox window of the MCLB may be necessary for some owner-draw functions. The MCLB creates the column listboxes with an ID, which is the same as the column number (1 based). To get the handle of the column 2 listbox, the application can use:

```
Col2Hwnd = WinQueryWindowFromID
           (MCLBHwnd, 2);
```

The window handle can be used to obtain a presentation space. The presentation space can then be used for `WinDrawText` (...`DT_QUERYEXTENT`) to accurately measure the size of text strings in the listbox.

DIRECT-MANIPULATION LISTBOX

The Direct-Manipulation ListBox (DMLB) is another extension to the basic PM listbox. The DMLB is a subclassing of a listbox and is unlike the MCLB, which is a composite custom PM control.

Applications often need to present a list of items in which the order of the items is significant. The listbox (or MCLB) control can be used to present such an ordered list. However, the user interface becomes more difficult if the application needs to provide a means for the user to specify a new ordering. Typically an application would have to resort to a series of pushbuttons and multistep user interactions to let the user reorder items in the list. The user interface is often clumsy and requires many user interactions to do significant reordering of the list.

Some applications also need to allow the user to move items from one listbox to another. For example, a database export program might present a list of fields in the database and let the user choose the fields to be exported. The exported list of fields appears in another listbox. Applications typically provide insert/delete or move buttons between the listboxes to let the user move items back and forth between them. Again, the user interface is not always intuitive and requires several interactions to control it.

The DMLB addresses these problems by adding easy-to-use drag-and-drop capability to the listbox (or MCLB) control. The user can drag a listbox item to a new position in the same list or to another list. The application controls what happens when the user drops the list item (the item is moved, copied, or deleted). The listbox can be scrolled while dragging by

moving the drag pointer above or below the listbox.

Figure 4 illustrates how a DMLB is typically used in a database application. The left listbox shows all the fields in the database, and the right list shows the fields selected for exporting. The user can drag-and-drop reorder the right list to change the order of exported fields. He/She can also drag fields from the left list and add (copy) them to the right one. Dragging a field from the right list to the left deletes it. With the DMLB, the user is able to easily choose database fields and order them for exporting without any additional dialog controls or complex user interactions.

PROGRAMMING THE DMLB

The Direct-Manipulation ListBox is not a custom control but rather a subclassing of the standard PM listbox. Thus, an application first creates the listbox in the usual fashion and then calls the `DMLBInitialize()` API to establish the subclass and initialize the DMLB instance data. Listing 4 shows how a listbox control is set up for direct manipulation during `WM_INITDLG` processing of a dialog.

The DMLB subclass, as implemented, will use the listbox window words (`QWL_USER`) for a pointer to DMLB-related instance data. The listbox window words are not available for application use. However, the DMLB reserves the first `PVOID` of its own instance data for application use. The application can use the first `PVOID` of the DMLB instance data if it needs to associate data with the listbox control.

```
HWND  ListHwnd;
PVOID *DMLBData;

WM_INITDLG:

  // Get handle of listbox control

  ListHwnd = WinWindowFromID(hwnd, ID_LISTBOX);

  // Setup subclassing for drag/drop support, resources
  // are bound to the EXE file

  DMLBInitialize(ListHwnd, NULLHANDLE);

  // Use DMLB instance data to store data we need to associate
  // with the listbox.  First PVOID of DMLB area is ours to use.

  DMLBData = WinQueryWindowPtr(ListHwnd, QWL_USER);
  *DMLBData = MyListData;

   //... remainder of dialog initialization ...//

   return (MRESULT)TRUE;
   }
```

Listing 4. *Setup of a drag-and-drop listbox.*

The drag-and-drop listbox support involves a simple message protocol between the source listbox (where the drag originates) and the target listbox (where the user drops the item). The source and target may be the same listbox. When a listbox item is dragged over a listbox window, the potential target window is "probed" to determine if an item can

WM_CONTROL Code	Description
LN_DMLB_DELETE	Item is about to be deleted (current selection).
LN_DMLB_DELETE_MOVE	Item is to be deleted for moving to another listbox.
LN_DMLB_REORDERED	An item was moved within the same listbox.
LN_DMLB_INSERT_MOVE	Item was inserted via drag: MOVE (current selection).
LN_DMLB_INSERT_COPY	Item was inserted via drag: COPY (current selection).
LN_DMLB_QRYDROP	A drop is about to occur on this listbox, return OK-to-drop flag and drop-mode flags (move/copy/delete).
LN_DMLB_CONTEXT	A context menu was requested on the listbox.

Table 2. *DMLB notification messages.*

be dropped on it. The target responds to the probe with various codes; it can refuse the drop, or if the drop is allowed, the target will decide if the drop is to be a move, copy, or delete of the original item. The mechanics of the dragging and probing is automatically done by the DMLB subclass procedure. The application only needs to respond to a few WM_CONTROL notification messages.

The complete list of DMLB notification messages is shown in Table 2. To implement a simple drag-and-drop re-ordering capability in the listbox, the application needs to respond only to the LN_DMLB_QRYDROP message. Listing 5 shows the processing of this message. The listbox will accept drops on itself for reordering items but refuses drops from any other listboxes.

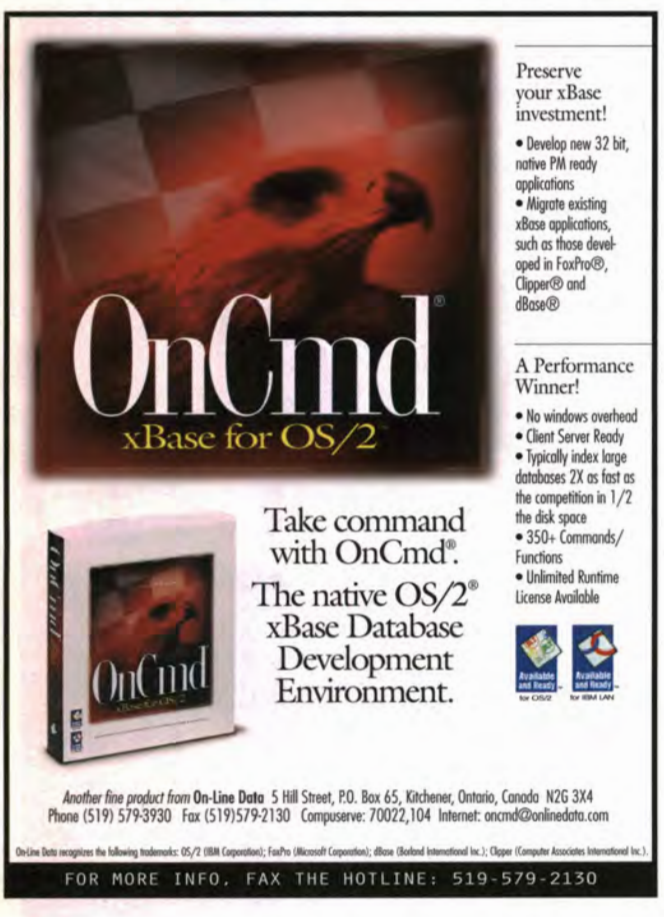
The owners of the source and target listboxes will receive messages when items are moved, copied, or deleted by drag-and-drop methods. The application can ignore these messages, if they are not of interest, or process them as required. For example, the application may use the listbox item handles to keep track of dynamically allocated data for each item in the listbox. The application will want to process the LN_DMLB_DELETE message to free the data when the item is deleted due to a drag-and-drop operation.

DMLB CONTEXT MENUS

Another useful feature of the DMLB subclass is the ability to notify the owner when the user requests a context menu. The DMLB subclass will send the owner an LN_DMLB_CONTEXT message along with the index of the item on which the context menu was requested. Therefore, the application can provide direct actions on listbox items via context menus. Figure 5 is a listbox with a context menu for adding new items, editing items, and deleting items.

DRAG-AND-DROP IMPLEMENTATION

The current DMLB subclass uses a private drag-and-drop message protocol. It is designed so that a drag of a listbox item over a listbox, which is not subclassed with the DMLB, will result in a "no-drop allowed" condition. The DMLB subclass handles capture of the



Circle Reader Service Number 14

Circle Reader Service Number 15

```
case WM_CONTROL:                    // Control notification
  switch SHORT1FROMMP(mp1) {
    case ID_LISTBOX:                // From drag/drop listbox
      switch (SHORT2FROMMP(mp1)) {

        case LN_DMLB_QRYDROP:
          // If dropping in same listbox, move the item.  If drop is
          // from some other listbox, refuse it.

          if (HWNDFROMMP(mp2) == WinWindowFromID(hwnd, ID_LISTBOX))
            return MRFROM2SHORT(TRUE, DROPMODE_MOVE);

          return MRFROM2SHORT(FALSE, 0);

          ...
```

Listing 5. *Processing* `LN_DMLB_QRYDROP` *message.*

Figure 5. *Listbox context menu.*

mouse pointer, sets the shape of the pointer, and probes the window under the pointer to determine suitability for dropping.

The DMLB subclass does not use the PM `DrgDrag()` API to implement the drag-and-drop functions. The advantage of this implementation is code simplicity (no complex drag data structures or message protocols), thereby allowing the owner of the listbox to respond to a very simple set of drag-and-drop messages. The disadvantage of this approach is that it precludes integration with the Workplace Shell. For example, a listbox item can't be dragged to a WPS folder. The intended scope of the DMLB subclass is only to support drag and drop on listbox class objects.

SUMMARY

The MCLB and DMLB provide Presentation Manager applications with intuitive and visually clean extensions to the common listbox control. Implementation in existing or new PM applications is made quick and easy by the use of the familiar listbox programming model. With a minimal amount of code or programmer training, an application can provide a sophisticated user interface with tabular data and/or drag-and-drop features. A complete toolkit, including full source code, online programming references, and working samples is available.

Mark McMillan *is an advisory engineer with IBM's Networking Software Division at Research Triangle Park, N.C. He received his B.S. at the University of Michigan and completed his M.S. in computer engineering at Duke University. McMillan is currently developing user interface components for IBM's Personal Communications software products.*

REFERENCES

OS/2 Warp Version 3.0 Presentation Manager Programming Reference, IBM G25H-7105.

Benge, Mark and Matt Smith. "A List Box Replacement," *OS/2 Developer*, (January/February 1994).

The MCLB and DMLB toolkit, including complete source code, samples, and online documentation, are available on CompuServe's OS2DF2 forum, *OS/2 Developer* section. Download file MCLB.ZIP.

An Internet-based Directory to Thousands of Products and Services

OS/2® & AIX developers, systems administrators, and users: the tools you need are just a few keystrokes away.

http://www.mfi.com/softwareguide

Thousands of product & service listings for IBM supported operating systems, continually updated, easy to search.

Sponsored by IBM®, *OS/2 Developer*, *OS/2 Magazine*, and *Software Development*.

OS/2® is a registered trademark of International Business Machines Corporation and is used by Miller Freeman, Inc. under license.

Find It on the Web!

REXX + dBase = RexxBase

The dBase Database Access For REXX

- Works with DBF, DBT, NDX and MDX files.
- Supports dBase III and IV file formats.
- Uses standard REXX API function calls.
- Comes with a GUI Front-End for Database Maintenance.

Find the shareware version ***RXBAS203.ZIP*** on your favorite BBS.

Use it with VisPro/Rexx,VX Rexx and GpfRexx

American Coders, Ltd. Post Office Box 97462 Raleigh, NC USA 27624
(919) 846-2014
Internet: joe@usacoder.rtp.nc.us CIS: 74150,2370
Only $95.00 + shipping & handling

Circle Reader Service Number 16

Send text messages to pagers directly from OS/2!

- Easy-to-use workplace shell application
- Includes a command line interface
- ***You can 'page-enable' your applications!***
- 32-bit, SOM-based code - ***Only $79.***

ChipChat® Wireless Communicator

from ChipChat Technology Group
Dearborn, Michigan USA & Koga, Fukuoka Japan

Phone 313-565-4000 http://www.chipchat.com

Circle Reader Service Number 17

TCP/2™

TCP/IP for OS/2®

Our customers say our tcp/ip is the best because it's faster, more complete, and reliable. Find out for yourself!

Supported: *all* released versions of OS/2 including 32bit API for 2.x and above; SLIP and/or PPP on COM1 and/or COM2; Ethernet; Token Ring; ARCNet; NDIS and ODI

(800) OS2-TCP2

Essex Systems, Inc.
One Central Street, Middleton, MA 01949

A Dan Lanciani Product

TCP/2 is a Trademark of DLD consulting. All other products and trademarks are the property of their owners. TCP/2 is Based in part on work done by the University of California Berkeley.

Circle Reader Service Number 18

FAX Developer Tools

From the developers of ***FaxWorks for OS/2***

Client/Server API and Printer Driver Toolkits
Support for LANs, up to 96 lines per CPU, and all popular fax hardware

Keller Group Inc.

Voice: (612) 429-7273 **Fax: (800) 329-3293**
Email: kgroup@ibm.net **Fax: (612) 653-1987**

Faxworks is a trademark of Global Village Communication.

Circle Reader Service Number 19

Opt-Tech Sort/Merge

High Performance Sort/Merge/Select Utility

Use as a stand-alone utility or call as a Subroutine. Many features including unlimited file size, multiple keys, record selection, duplicate elimination, summing and more!

Call for free brochure

Available for:
OS/2 or Unix $249
DOS or Windows $149

Opt-TechData Processing
P.O. Box 678
Zephyr Cove, NV 89448
Phone (702) 588-3737
Fax (702) 588-7576

Circle Reader Service Number 20

The easiest to use, most powerful 32 bit OS/2 PM IPF Language Editor...

New! IPF Editor 2.0

SRP: $195

- **Designed for intuitive use by anyone**
- **Supports all IPF Tags, with simple index creation, easy multiple-window creation, and drag-and-drop panel ordering**
- **Generate C program and Resource source to add help to applications in minutes**
- **Import WordPerfect, Describe and RTF**
- **Preview IPF output without compiling**
- **Create all hypertext links automatically**
- **Sound Support and Spell Check**
- **Complete on-line help and documentation**
- **Optional multi-user nework version**

Orders: 1-800-IPF-7622

Information: (360)428-5025 on compuserve at 70410,2416

Perez Computing Services
4725 Monte Vista Place
Mt Vernon, WA 98273

Circle Reader Service Number 21

PrntScrn Screen Image Utility by MITNOR Software

4 Integrated Utilities for 1 Economical Price

* Screen Printing/Screen Capture
8 Capture methods/7 File Formats
Print to Local or LAN attached printers
* Clipboard Viewer with Import, Export & Printing capabilities
* Screen Saver with 9 Dynamic Display Types
* Date & Time Information Bar

Available From
Indelible Blue 1-800-776-8284

Circle Reader Service Number 22

Network Remote Control And Monitoring

Help Desks Automated Agent Monitoring

PM/Pirate

Named Pipes
TCP/IP
Lowest network load
Record and Playback
All PM Video Modes
No Special Drivers
FAST!
Automated Cycling

$89.00 + s&h

SOFTEL

Digital Communications Solutions
2050 North Towne Ct. NE Suite 7
Cedar Rapids, IA 52402

Softel also provides:
In and Outbound call center applications -
Voice information systems
Predictive dialing
Voice Mail
Internet Service provider and Client Software.
Presentation Manager Development and Consulting

(800) 701-7205

Circle Reader Service Number 23

TRACK 'em DOWN!™

SOFFRONT® TRACK

Track:
- Bugs, Change Requests
- Projects, Code Changes
- Customer Calls and more

Customize:
- Reports
- Queries
- Forms & fields

Automatic Notification: Nothing falls through the cracks
Integration with Version Control: Link files to bugs & bugs to files
Multiple Linked Databases: Track bugs, customers and more
Industry Standard Databases: Access from other applications

SOFFRONT Software Inc.

FREE *working* DEMO!

1-800-763-3766 / 408.263.2703
238 S. Hillview Drive, Milpitas, CA 95035 Fax: 408.263.7452

Circle Reader Service Number 24

Jagre SmartLink and Lotus Notes: A Winning Combination!

File Open
File Save
E-mail
Fax
Pager
OCR
Audio
Video
iRetriever
Help

In today's computer world, communicating effectively and quickly could be the difference between your company being a dead company or being a quick company.

Using Lotus Notes within your company, gives your company the ammunition it needs to be a quick company.

Now, Jagre, Inc. provides your company with **the** competitive advantage to being -- and staying -- a quick company: **Jagre SmartLink**.

Jagre SmartLink is a set of Lotus Notes-enabled tools grouped together in one place to empower your IBM OS/2-based desktop applications to communciate effectively and quickly with Lotus Notes for Document Management, E-mail, Fax, Pager, OCR, Voice, Video, and User-based Agents to gets tasks done quickly.

With Jagre SmartLink, the powerful features of Lotus Notes are now a button click away!

Click on any Jagre SmartLink tool for instant access to Lotus Notes. Now, you can use Lotus Notes and the Lotus Notes Companion Products with your desktop applications to communicate quickly with someone down the hall or across the world!

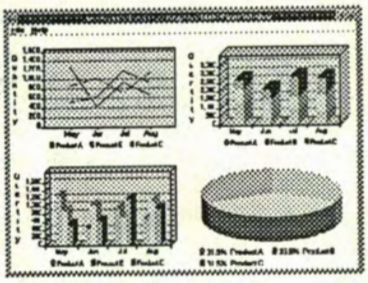
Now, your ammunition arsenal is complete with Jagre SmartLink and their arsenals are not!!

Jagre SmartLink

For a free demonstration copy of Jagre **SmartLink to discover how it can help your company be a quick company, not a dead company, call or write Jagre, Inc.** at (617)424-8302, 102 Gainsborough St. #202E Boston, MA 02115.

Jagre SmartLink

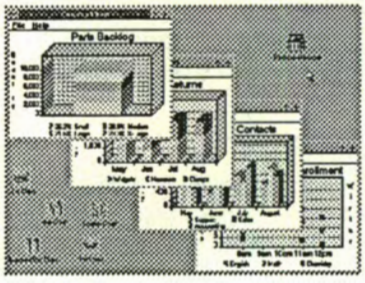
Circle Reader Service Number 25



It's easy to integrate business graphics into your OS/2 PM or Windows 3,NT,95 applications

Developer's Business Graphics Toolkit

- **High-level APIs**
- **Tactile feedback**
- **Real-time charting**
- **16-bit and 32-bit**
- **Dynamic Link Library**
- **Positive/negative data**
- **Printer/metafile support**
- **25+ chart types**



The Crossley Group
P.O. Box 921759
Norcross, GA 30092
USA (770) 751-3703
Int'l 44-932-844-281
(demo available)

Circle Reader Service Number 26

What are you waiting for?

Right now you could be reaching more than 36,000 product-buying OS/2 Developers!

Call now for details

Clair Bright
(415)905-2544
cbright@mfi.com

VisualAge & PARTS

Spreadsheets, business graphics, tables, and other powerful controls are are easy to add to your VisualAge™ and PARTS™ applications.
WidgetKit™/Professional provides spreadsheets, tables, virtual spreadsheets, spreadsheet designer, load and save, printing, and more.
WidgetKit/Business Graphics has versatile graphs and charts.
Subpanes/V for PARTS has a powerful table pane, columnar list box, heirarchical list box, 3-D frames, 3-D and bitmap buttons, and more.
No royalties for end-user applications, 90 days free support, and 30-day money-back guarantee. Call or fax for free info.

Objectshare Systems, Inc. **v: 408.970.7280 f: 408.970.7282**
5 Town & Country Village, Suite 773, San Jose, CA 95128

Circle Reader Service Number 27

DeskMan/2™

Building A Better Desktop℠

The Award-Winning, Multi-Function Desktop Enhancements for OS/2®

Presenting the DevTech Family of Productivity Tools for the OS/2 User.

DeskMan/2™

- *Manage, secure, and customize your OS/2 environment like never before!*
- *Provide personalized desktops for each user.*
- *Switch between as many as 81 "virtual desktops"—like having up to 81 monitors that all fit right on your desk!*
- *Save and restore entire system configurations.*

Best selling ***DeskMan/2*** is the award-winning enhancement for all personal and corporate OS/2 users. This comprehensive, flexible, one-of-a-kind suite of multi-function tools makes it easy to use and administer OS/2, from the standalone PC to the LAN! Install, configure, backup, secure, and otherwise manage desktops locally or remotely. Enhance productivity with virtual desktops and Workplace Shell Extensions. Authorize access to individual ***DeskMan/2*** and Workplace Shell features on a user by user basis. Take complete control of the WPS via a simple drag & drop interface, or a powerful API (including full support for REXX, CID and LAD/2). Review after review concludes that ***DeskMan/2*** is a "must have" for all OS/2 users. IBM even features ***DeskMan/2*** in its own "Red Book" on WPS configuration. Why not find out for yourself?

The DeskMan/2™ Productivity Pack

*Special Editions created exclusively for the Productivity Pack.

- ***DeskMan/2 v1.51b plus*** *FREE upgrade to* ***DeskMan/2*** *v2.0 (when available).*
- ***DCF/2 Lite™****- The "on the fly" data compression for OS/2.*
- ***The Graham Utilities for OS/2, Limited Edition**** *- Special selection of disk utilities.*
- ***Relish® v2.12*** *- Award-winning calendar, reminder & personal planner for OS/2.*
- ***CPU Monitor Plus™ Special Edition**** *- professional OS/2 performance monitor and analysis package.*

Current DeskMan/2 users can upgrade to the Productivity Pack. Call for details.

Take the award-winning ***DeskMan/2*** product; include features previously available only to corporate users, such as enhanced security and Workplace Shell auditing; add a leading edge selection of invaluable OS/2 utility products; and top the whole thing off with an incredible array of upgrade offers—that is the essence of the wildly successful ***DeskMan/2 Productivity Pack!*** This is one-stop-shopping for all your essential OS/2 needs, and has been called nothing less than "the OS/2 Survival Kit!"

DCF/2™

- ***All new version!***
- *Now twice as fast and ultra reliable!*
- *Real world compression > 2.5:1.*
- *No pre-allocation required - dynamic space management.*
- *Unique disk server architecture.*

DCF/2 brings the power of a disk server architecture, usually found only on expensive mainframes and high-end minicomputers, down to your OS/2 desktop or notebook machine. Use HPFS without having to repartition or reformat your existing hard drive; DCF/2's unique architecture allows it to create virtual HPFS drives anywhere that OS/2 can create a file: FAT partitions, floppy disks, removable hard drives, Novell servers, LAN Server volumes, you name it—DCF/2 volumes go anywhere! Plus, DCF/2 volumes grow and shrink as needed so you don't need to pre-allocate disk space! Copy entire HPFS virtual drives between computers as easily as copying a single file. New "After Image Journaling" technology brings even more mainframe power to the PC, and gives DCF/2 the most reliable data storage you can buy for OS/2—even surviving power failures!

The Graham Utilities™

- *Comprehensive suite of disk, file and general purpose utilities.*
- *The* ***only*** *LAN compatible disk recovery tools for OS/2!*
- *Edit, manage and repair local and remote disk drives.*
- *Restore corrupt partitions, recover lost data, diagnose system problems.*
- ***Much, much more — 50+ modules in all!***

The Graham Utilities is the most comprehensive suite of disk, file and general purpose utilities available for OS/2. This powerful suite contains tools that range from system diagnostics, to file and disk management, to helping you deal with the Internet. Recover lost data, undelete files, edit, manage and repair disk drives — even remote disk drives on LAN Server, LAN Manager, Lantastic and Novell Netware! The package includes an extensive 300+ page printed manual detailing all aspects of the product and its use, complete on-line documentation for easy access, and the unique Graham Integrator that literally walks you through the entire suite. A "must have" for every OS/2 user!

"I can't imagine being without it."
Dave Barnes, IBM

"DeskMan/2 dramatically improves the ability of corporations and users to get the most out of OS/2. DeskMan/2 helps administrators restrict and control what users are doing and provides robust backup/restore features and seamlessly enhances the features of the WPS"
Dr. Mike Kogan: Lead Architect of OS/2 2.0 Author: The Design of OS/2

Circle Reader Service Number 28

DevTech
Development Technologies, Inc.
Software Development & Technology Transfer
308 Springwood Road, Forest Acres, SC 29206-2113
Voice: (803) 790-9230 • Fax: (803) 738-0218

Copyright 1995 Development Technologies, Inc. All rights reserved. DeskMan, DeskMan/2, VUEMan, VUEMan/2, DM2Image, DCF/2 and DCF/2 Lite are trademarks of Development Technologies, Inc. Building A Better Desktop is a service mark of Development Technologies, Inc. Workplace Shell is a trademark of International Business Machines Corporation. IBM and OS/2 are registered trademarks of International Business Machines Corporation. Relish is a registered trademark of Sundial Systems Corporation. The Graham Utilities is a trademark of WarpSpeed Computers. CPU Monitor Plus is a trademark of BonAmi Software Corporation. Other trademarks belong to their respective companies.

EDITORS' CHOICE AWARD OS/2 MAGAZINE 1994

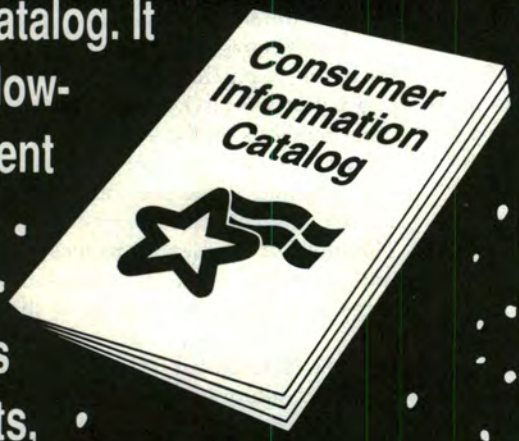
Best of OS/2 1994 OS/2 Inside Reader's Choice Award

Ready for OS/2 WARP

Ready for OS/2 WARP LAN Server

MAY THE SOURCE BE WITH YOU

Don't let the dark forces of ignorance defeat you. Right in this galaxy, you can tap into the source -- the free Consumer Information Catalog. It lists free and low-cost government publications on cosmic topics such as federal benefits, jobs, health, housing, educating your children, cars, and much, much more. So dispel the darkness and send for the source. Write today to Pueblo, Colorado for the free Consumer Information Catalog. Just send your name and address to :



Consumer Information Center
Department Source
Pueblo, Colorado 81009

A public service of this publication and the Consumer Information Center of the U. S. General Services Administration

ADVERTISER INDEX

Reader Service	Company Name	Page
16	American Codersx	54
17	ChipChat-Cawthon	54
15	Cirrus Technology	52
26	Crossley Group	55
28	Development Technologies	56
18	Essex	54
1	Hockware	1
5,31	IBM	10-11,COV3
13	Indelible Blue	51
—	Inmark	COV2
25	JAGRE	55
19	Keller Group	54
29	Lieberman	63
11	MHR	34
4	Microedge	9
22	Mintnor	54
10	OS/2 Express	29
9	Object People	23
27	Objectshare	55
14	On-Line Data	52
20	Opt-Tech	54
21	Perez Computing Services	54
2	Programmer's Paradise	2
7	Quadron	18
8	Rimstar	19
6	Segue	17
24	Soffront	55
30	Softbridge	63
23	Softel	55
32	Softouch	COV4
12	System FX	45
3	Watcom	4

OpenDoc gives developers a whole new way to create applications. Here's a great way to leverage multimedia technology.

By SCOTT BROUSSARD

Accessing Multimedia Data from OpenDoc

Scott Broussard

Now that OpenDoc has given application developers a new component-based platform in which to develop applications, and multimedia is starting to find its place in mainstream applications, the first thing a developer needs to do is understand how to leverage multimedia technology, such as digital audio, and embed it in an OpenDoc compound document.

ABOUT STORAGE UNITS

OpenDoc uses a compound document technology, called Bento, to store the data of various types of component parts in a document. The OpenDoc run time reserves an area of the compound file for use by each part, called a storage unit. The OpenDoc programming interfaces provide a class called `ODStorageUnit` that encapsulates the component's access to the Bento file. Within each storage unit, there is a set of properties, where each property has a set of values. Listing 1 shows the architecture of the storage unit.

Properties and Values have names that are strings. These names are frequently ISO Strings. ISO Strings are scoped names that contain various substrings delimited by colons to ensure that they are unique across any domain in which they need to be unique. These substrings generally contain a company or organization name. For example, `"IBM:SamplePart:Kind"` is an ISO string that could be used as a Property or Value name.

Access to data within a storage unit is similar to access to data within a traditional data file, although some key differences do exist.

The `ODStorageUnit` class needs to maintain control over its internal structures, and therefore, provides `Lock()` and `Unlock()` functions to ensure that several threads don't try to access the same storage unit concurrently.

The `ODStorageUnit` class has the notion of Focus-ing a particular Property and Value pair with the `Focus()` method. This allows the other methods on the storage unit to operate on a particular segment of data. Only one Property and Value pair can have the focus within the storage unit at one time. Therefore, it is essential to lock the storage unit while accessing data, so that no other threads change the context of the storage unit.

The `ODStorageUnit` class allows the application to determine if a particular Property or Value exists without causing an error.

Data can be read from the storage unit with the `GetValue()` method and written with the `SetValue()` method. These methods access the data at the currently set offset, which can

```
+-Storage Unit-------------------------+
| +-Property-------------------------+ |
| | +-Value--------------------------+ | |
| | | application data               | | |
| | +--------------------------------+ | |
| | +-Value--------------------------+ | |
| | | application data               | | |
| | +--------------------------------+ | |
| +-----------------------------------+ |
| +-Property-------------------------+ |
| | +-Value--------------------------+ | |
| | | application data               | | |
| | +--------------------------------+ | |
| | +-Value--------------------------+ | |
| | | application data               | | |
| | +--------------------------------+ | |
| +-----------------------------------+ |
| ...                                   |
+---------------------------------------+
```

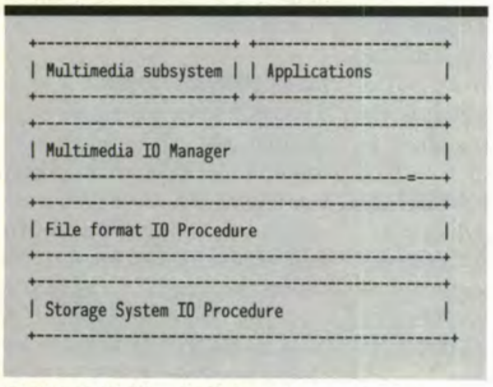
Listing 1. *Storage unit architecture.*

be modified with `SetOffset()`. It is important to note that when the Property loses the focus, the offset is reset.

ABOUT MULTIMEDIA IO

If an application component, such as a personnel information manager record that contained embedded audio, wanted the multimedia subsystem to access the data directly, without having to shuffle the data back and forth from a simple file and a storage unit of a compound document, the application would want to use a Multimedia IO (MMIO) Storage System IO Procedure.

The multimedia programming interfaces are set up to primarily deal with files that contain the common multimedia data formats, such as .WAV or .AVI. Fortunately, the MMIO Manager is organized into both File Format IO procedures, which manipulate the layout of the data, and Storage System IO procedures, which manage where the data is stored. This allows data to be stored in either files or memory, or in the case of OpenDoc, in Properties of a storage unit that are reserved from the component.



Listing 2. *Multimedia IO Procedure architecture.*

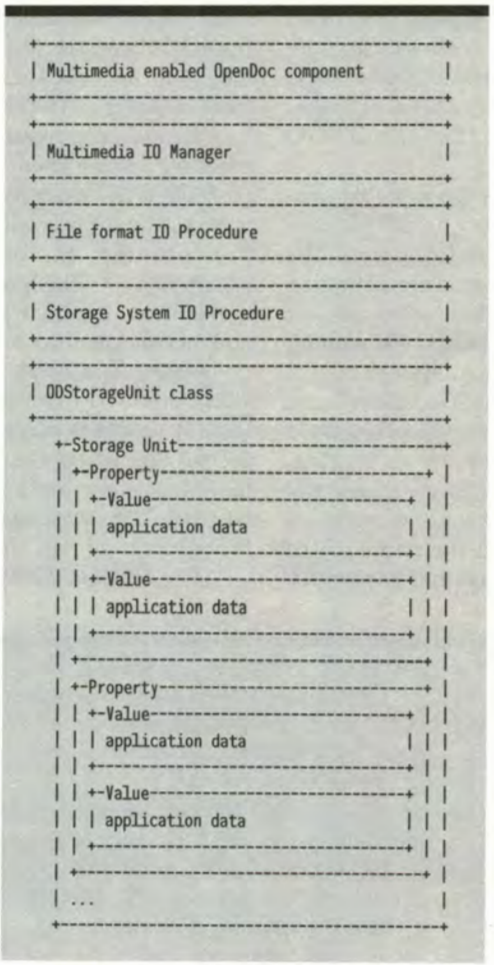
Listing 2 shows how the MMIO Manager layers its IO Procedures to allow the data to be stored anywhere but still uses the same code to manipulate the layout of the data.

By writing a MMIO Storage System IO Procedure, the OpenDoc component handler that is using multimedia data types within a compound document can use the existing multimedia subsystem transparently without making any modifications or doing any of the basic work itself. As in Listing 3, it can implement an IO Procedure that uses the `ODStorageUnit` object to store and retrieve the multimedia data from within the compound document.

USING THEM TOGETHER

In order to begin thinking about an OpenDoc Storage Unit IO procedure for MMIO, it is necessary to understand that the multimedia programming interfaces are used to dealing with media object names, which are usually file names.

The MMIO manager identifies which Storage System IO Procedure should be used based on the syntax of the media object name. Simple file names and paths are directed to the DOS Storage System IO Procedure, which uses OS/2 File System APIs to implement the necessary functions. If the media object name is a file name with a '+' sign and an element name, then the BND (Bundled or compound) file Storage System



Listing 3. *Multimedia OpenDoc Storage Unit architecture.*

```
FOURCC+ssssssss+PropertyName+ValueName.Extension
```

Listing 4. *Sample media object naming syntax.*

IO Procedure is used. The Bundled file is a legacy compound file format that is built into MMIO, but it shouldn't be confused with the OpenDoc compound file, as they are different things. If no media object name at all is given, and a memory block is provided, then a memory file will be used.

The media object name is the key to making the MMIO Storage System IO Procedure for OpenDoc Storage Units work. The media object name must contain all of the information necessary to access the object properly, which includes ensuring that the MMIO subsystem correctly identifies the OpenDoc IO Procedure when appropriate and that the correct Storage Unit, Property, and Value names are accessed. The programmer has some freedom in determining the syntax of the media object names that the Storage System IO Procedure will use.

The MMIO Manager uses a naming convention to help expedite the identification of the Storage System IO Procedure. A dot, followed by the FOURCC (Four Character Code) used by the IO Procedure and a '+' sign, will ensure that the correct IO Procedure is used.

Listing 4 shows the possible syntax of the media object name that contains the necessary information. The components of the name are delimited by '+' signs because they are consistent with the legacy compound file naming convention but are not required.

Because the OpenDoc component handler is passed an `ODStorageUnit *` when it is created for storing its data, and a storage unit may or may not have a valid name, it is necessary to include the pointer in the name of the media object. It is possible to pass the pointer in the form of a string embedded as a piece of the compound name. The Property and Value names should also be included in the compound name.

Finally, the appropriate file extension for the data should be appended to the Value name, so that the Media Device Manager (MDM) can properly route the open request to the appropriate MCI Driver. Combining the file extension (normally be used for the data) to the Value name and stored in the document is preferable. If, for example, the data is a .FLI or .FLC animation file, which are normally silent, the File Format IO Procedure in OS/2 looks for a corresponding .WAV file to use as an audio track. If there is another Value that has the same name, but with a .WAV extension, then it could be used for the audio track.

IMPLEMENTING THE STORAGE SYSTEM IO PROCEDURE

The MMIO Storage System IO Procedure must implement the set of commands shown in Table 1.

Since a pointer embedded in the media object name is being passed to the IO Procedure, the IO Procedure may want to validate it somehow before accessing it. The OpenDoc component handler can register valid storage units by adding them to a list. When an `MMIOM_OPEN` or an `MMIOM_IDENTIFYFILE` occurs, the IO Procedure can validate the pointer as being a valid `ODStorageUnit *`. The Storage Unit is removed from the list when the object is no longer accessible.

`MMIOM_BEGINSTREAM` and `MMIOM_ENDSTREAM` are used to indicate when playback or record is occurring. This gives the IO Procedure the opportunity to request a certain level of service from a network. The `ODStorageUnit` MMIO IO Procedure need only return 0.

The `MMIOM_GETFORMATINFO` message provides capability information about the storage unit to the MMIO manager. The IO Procedure should indicate that it is a Storage System IO Procedure rather than a File Format IO Procedure. It should also indicate that it can `read/write/seek/readwrite` untranslated.

The `MMIOM_GETFORMATNAME` message simply returns a user-readable name for the Storage System IO Procedure, however, no applications that I know of actually display Storage System IO Procedures.

For `MMIOM_IDENTIFYFILE`, the IO Procedure should validate the format of the name and return `MMIO_SUCCESS` if it is valid or `MMIO_ERROR` if it is not valid. At this point, the `ODStorageUnit *` shouldn't be validated against the list, but rather during the `MMIOM_OPEN` processing. If the IO Procedure fails to identify a name that is in fact intended for the `ODStorageUnit` IO Procedure, it will then probably be identified by the DOS IO Procedure, and the file system will surely report an `ERROR_INVALID_NAME` error.

The IO Procedure will need to have an instance data structure that contains the open flags, the `ODStorageUnit *`, the Property name, and the Value name. These instance data structures should be linked together into a list so that multiple opens of the same Property/Value pair can be detected and access control emulated (like the File System).

IMPLEMENTING OPEN

The `MMIOM_OPEN` processing is the most complicated. The errors returned from the IO Procedure are important so that the use of this IO Procedure is transparent to existing software. This means that OpenDoc `ODStorageUnit` error conditions should be translated to MMIO errors or OS/2 File System errors. File system errors are frequently put into the `ulErrorRet` field of the `MMIOINFO` structure.

When processing the `MMIOM_OPEN` message, the IO Procedure should validate the name to ensure that it has all of the required pieces of information. If it isn't a valid name, the procedure should return `MMIOERR_INVALID_FILENAME`.

The procedure should then determine if there are any other instances being accessed and check the open flags of both instances to determine if they are compatible. If they are not compatible, then return `MMIOERR_INVALID_ACCESS_FLAG`.

The instance data structure can then be initialized. Typically, the

`MMIOM_OPEN`	Open the specified media element
`MMIOM_CLOSE`	Close the media element
`MMIOM_READ`	Read a block of data
`MMIOM_WRITE`	Write a block of data
`MMIOM_SEEK`	Seek to a specified offset in the data
`MMIOM_IDENTIFYFILE`	Identify the media element name
`MMIOM_GETFORMATINFO`	Return information about the IO Procedure
`MMIOM_GETFORMATNAME`	Return the name of the IO Procedure
`MMIOM_BEGINSTREAM`	Begin streaming the data
`MMIOM_ENDSTREAM`	End streaming the data

Table 1. *Required MMIO messages for Storage System IO Procedure.*

```
ULONG EXPENTRY SUIOProcOpen( PMMIOINFO pmmioinfo,
                             LONG lParam1,
                             LONG lParam2)
{
  ULONG size = 0;     // size of property
  ULONG rc;           // return code
  PSZ pszFileName     // file name
  PSUINFO pSUInfo;    // Storage Unit object instance information
  BOOL fLock = FALSE; // Storage Unit locking
  Environment * ev = somGetGlobalEnvironment();

  if (!pmmioinfo) return (MMIO_ERROR);
  pszFileName = (PSZ)lParam1;  // get the filename from
                  parameter.
  if (!pszFileName) return (MMIOERR_INVALID_FILENAME);

  // validate name (basically identify the object)

  pSUInfo = ValidateName (pszFileName);
  if (!pSUInfo) return (MMIOERR_INVALID_FILENAME);
  ReleaseName (pSUInfo);

    // Get access to the object

  pSUInfo = AccessName (pszFileName, pmmioinfo->ulFlags);
  if (!pSUInfo) return (MMIOERR_INVALID_ACCESS_FLAG);
  pmmioinfo->aulInfo[1] = (ULONG) pSUInfo;  // save it for later

  // complete the open

  if (pmmioinfo->ulFlags & MMIO_DELETE) {

     pSUInfo->key = pSUInfo->pStorageUnit->Lock (ev, 0);
     if (!IsException(ev)) {
        if (pSUInfo->pStorageUnit->Exists (ev,
                           pSUInfo->szPropertyName,
                           pSUInfo->szValueType,
                          0)) {
          pSUInfo->pStorageUnit->Focus(ev,
                pSUInfo->szPropertyName,
                kODPosFirstSib,
                 pSUInfo->szValueType,
                0,
                kODPosFirstSib);
          if (!IsException(ev)) {
               pSUInfo->pStorageUnit->Remove(ev);
               pSUInfo->pStorageUnit->Unlock (ev, pSUInfo->key);
            } else {
               pmmioinfo->ulErrorRet = ERROR_FILE_NOT_FOUND;
               pSUInfo->pStorageUnit->Unlock (ev, pSUInfo->key);
               ReleaseName (pSUInfo);
               return (ERROR_FILE_NOT_FOUND);
            }
          } else {
            pmmioinfo->ulErrorRet = ERROR_FILE_NOT_FOUND;
            pSUInfo->pStorageUnit->Unlock (ev, pSUInfo->key);
            ReleaseName (pSUInfo);
            return (ERROR_FILE_NOT_FOUND);
          }

        } else {
           pmmioinfo->ulErrorRet = ERROR_ACCESS_DENIED;
           pSUInfo->pStorageUnit->Unlock (ev, pSUInfo->key);
           ReleaseName (pSUInfo);
           return (ERROR_FILE_NOT_FOUND);
        }

        ReleaseName (pSUInfo);
        pmmioinfo->aulInfo[1] = 0;
        return (0);
```

```
  } else if ( pmmioinfo->ulFlags & MMIO_CREATE ) {

     // Remove the Property / ValueType if there

     if (pSUInfo->pStorageUnit->Exists (ev,
                           pSUInfo->szPropertyName,
                           pSUInfo->szValueType,
                          0)) {
        pSUInfo->key = pSUInfo->pStorageUnit->Lock (ev, 0);
        if (!IsException(ev)) {
           pSUInfo->pStorageUnit->Focus(ev,
                 pSUInfo->szPropertyName,
                 kODPosFirstSib,
                 pSUInfo->szValueType,
                 0,
                 kODPosFirstSib);
           if (!IsException(ev)) {
              pSUInfo->pStorageUnit->Remove (ev);
           }
        }
        pSUInfo->pStorageUnit->Unlock (ev, pSUInfo->key);
     }

     // now create it new property

     pSUInfo->key = pSUInfo->pStorageUnit->Lock (ev, 0);
     if (!IsException(ev)) {
        pSUInfo->pStorageUnit->AddProperty(ev,
                 pSUInfo->szPropertyName);
        if (!IsException(ev)) {
           pSUInfo->pStorageUnit->AddValue (ev,
                 pSUInfo->szValueType);
           if (!IsException(ev)) {
              pSUInfo->pStorageUnit->Unlock (ev, pSUInfo->key);
           } else {
              pSUInfo->pStorageUnit->Unlock (ev, pSUInfo->key);
              ReleaseName (pSUInfo);
              pmmioinfo->ulErrorRet = ERROR_INVALID_NAME;
              return (ERROR_INVALID_NAME);
           }
        } else {
           pSUInfo->pStorageUnit->Unlock (ev, pSUInfo->key);
           ReleaseName (pSUInfo);
           pmmioinfo->ulErrorRet = ERROR_INVALID_NAME;
           return (ERROR_INVALID_NAME);
        }
     } else {
        pSUInfo->pStorageUnit->Unlock (ev, pSUInfo->key);
        ReleaseName (pSUInfo);
        pmmioinfo->ulErrorRet = ERROR_INVALID_NAME;
        return (ERROR_INVALID_NAME);
     }

     pmmioinfo->lLogicalFilePos = 0;

  } else {                    /* standard open */

     pSUInfo->key = pSUInfo->pStorageUnit->Lock (ev, 0);
     if (!IsException(ev)) {
        if (pSUInfo->pStorageUnit->Exists (ev,
                           pSUInfo->szPropertyName,
                           pSUInfo->szValueType,
                                    0)) {
           pSUInfo->pStorageUnit->Focus(ev,
                 pSUInfo->szPropertyName,
                 kODPosFirstSib,
                 pSUInfo->szValueType,
                 0,
                 kODPosFirstSib);
           if (!IsException(ev)) {
              size = pSUInfo->pStorageUnit->GetSize(ev);
```

Listing 5 *Open processing example.* *(Continued on page 62.)*

```
            pSUInfo->pStorageUnit->Unlock (ev, pSUInfo->key);
          } else {
            pSUInfo->pStorageUnit->Unlock (ev, pSUInfo->key);
            ReleaseName (pSUInfo);
            pmmioinfo->ulErrorRet = ERROR_FILE_NOT_FOUND;
            return (ERROR_FILE_NOT_FOUND);
          }
        } else {
          pSUInfo->pStorageUnit->Unlock (ev, pSUInfo->key);
          ReleaseName (pSUInfo);
          pmmioinfo->ulErrorRet = ERROR_FILE_NOT_FOUND;
          return (ERROR_FILE_NOT_FOUND);
        }
      } else {
        pSUInfo->pStorageUnit->Unlock (ev, pSUInfo->key);
        ReleaseName (pSUInfo);
        pmmioinfo->ulErrorRet = ERROR_ACCESS_DENIED;
        return (ERROR_ACCESS_DENIED);
      }

      if (size == 0) {
        // This will cause the audio mcd to call us back
                    again with a create
        pmmioinfo->ulErrorRet = ERROR_FILE_NOT_FOUND;
        ReleaseName (pSUInfo);
        return (ERROR_FILE_NOT_FOUND);
      }

      if (pmmioinfo->ulFlags & MMIO_APPEND) {
        pmmioinfo->lLogicalFilePos = size;
      } else {          // normal open
        pmmioinfo->lLogicalFilePos = 0;
      }

    }

    return (0);

}
```

Listing 5. *Open processing example. (Continued from page 61.)*

pointer is stored in the `aulInfo[1]` field of the `MMIOINFO` structure, which is passed to the IO Procedure on each subsequent call by the MMIO Manager.

During the processing of `MMIOM_OPEN` the open flags may contain, `MMIO_DELETE`, which indicates that the Value should be deleted. This is done by calling `Lock()`, `Exists()`, `Focus()`, `Remove()`, and `Unlock()` on the `ODStorageUnit` object. These are SOM 2.x IDL methods, and it is very important to always check the Environment for exceptions if they occurred, so that they can be translated to MMIO Errors and returned properly. The common errors here are `ERROR_ACCESS_DENIED` or `ERROR_FILE_NOT_FOUND`.

`MMIO_CREATE` may also be passed as an open flag. In this case, a new Property/Value pair should be created by calling `Lock`; `Exists()` to determine if it already exists; and `Focus()`, `AddProperty()`, `AddValue()`, and `UnLock()`. `ERROR_INVALID_NAME` or `ERROR_ACCESS_DENIED` is returned in the `ulErrorRet` field if something then fails here.

The `MMIOM_OPEN` could also be a standard open, with or without the `MMIO_APPEND` flag. If the `MMIO_APPEND` flag is set, the logical file position should be initialized to the length, otherwise to 0. The length of a particular Focus-ed Value can be determined with the `GetSize()` method. The `lLogicalFilePos` field of the `MMIOINFO` structure defines the file position.

One anomaly occurs during open processing: if the IO Procedure is asked to open a Property/Value pair that exists but has zero length, it should be treated as if the property doesn't exist. All multimedia files have headers. If the `mmioGetHeader()` fails, the subsystems check for Extended Attributes to determine if the file is a template. Since Extended Attributes are not supported by any Storage System IO Procedures, the File Format IO Procedures directly use file system APIs. By treating a zero length Property as if it doesn't exist, the multimedia File Format IO Procedures will create it and initialize it properly.

Listing 5 shows how the Open processing would be implemented.

BASIC READ/WRITE PROCESSING

The remaining IO functions are more straightforward. `MMIOM_READ` is processed by calling `Lock()`, `Focus()`, `SetOffset()` to the most recent seek offset, `GetValue()` to get block of data, and `Unlock()`.

`MMIOM_WRITE` is processed by calling `Lock()`, `Focus()`, `SetOffset()` to the most recent seek offset, `SetValue()` to set the block of data, and `Unlock()`. On the write, it is important to check the bytes actually written, and if it is less than the write attempt, return the `MMIO-ERR_CANNOTWRITE` error. This indicates that the disk is full.

`MMIOM_SEEK` just updates the `lLogicalFilePos` field in the `MMIOINFO` structure. This offset will be used on the next read, write, or seek operation.

`MMIOM_CLOSE` just releases the instance data structure.

Because this IO Procedure uses a pointer in the media object name, it's probably not a good idea for the application to store these media object names persistently anywhere. Also, since only your application has knowledge of this IO Procedures media-naming syntax, it's a good idea to register the IO Procedure only for the application's process and not for the whole system. This is easily done by using the `mmioInstallIOProc()` function of MMIO.

With a Storage System IO Procedure that makes OpenDoc Storage Unit calls, Multimedia data can be transparently integrated into most any OpenDoc component and embedded into any OpenDoc aware application. This gives application developers much more flexibility and more opportunity to combine their multimedia-enabled software with other software components to yield unique solutions.

ADVANCED TOPICS

Another topic to consider when designing a Storage System IO Procedure for OpenDoc is whether the storage unit needs to be accessed by more than one process. For example, drag-and-drop or clipboard support in the OpenDoc component handler may want to avoid copying large amounts of data, such as a video or large audio annotation, so it might choose to pass a file name or media object name where the data can be accessed by the receiver. This problem would require letting the owner process service requests to access the data by using some custom mechanism like PM message passing between processes or possibly using Distributed SOM (DSOM) to access the object. If multiple processes are involved, its a good idea to include the process ID of the process that owns the `ODStorageUnit` object in the name of the media object. Allowing only read access would probably be a good idea from another process.

Another issue is undo/redo support in the OpenDoc component handler. Typically, before chaining some internal data of a component, the han-

dler will put the old data into an undo stack, so that the modification can be undone at a later point in time. If you are using the multimedia subsystem to directly put data into the storage unit, the handler will want to know just before the data is written that the data is going to be modified, allowing it to create an undo record. Large amounts of data are a bit problematic for undo, however, the `ODStorageUnit` IO Procedure could provide a callback to the component handler when `MMIOM_WRITE` or `MMIOM_OPEN` with `MMIO_CREATE` is called.

OTHER WAYS

Of course, other ways do exist to integrate multimedia data into OpenDoc component handlers. OS/2 provides basic multimedia component handlers for audio, video, image, and MIDI. These handlers provide some of the more common functions that a user would want to do with sampled audio, digitized video, scanned images, or computerized music. These component handlers could be embedded into a more specialized component programmatically as well as by the users, giving them a common interface for dealing with these data types and freeing up the programming to provide more of what they are good at.

Scott Broussard *is an advisory programmer in OS/2 multimedia development and is responsible for designing and developing multimedia user interfaces. He is currently the lead architect for the OS/2 Multimedia Workplace Shell/OpenDoc team. Broussard can be reached at Scott_Broussard@bocaraton.ibm.com.*

REFERENCES

BM Multimedia Presentation Manager Toolkit/2 Programming Reference, (IBM order number S41G-2920).

"IBM OpenDoc," OS/2 Specific Class Reference, *OS/2 Developer's* Connection CD, August 1995.

IBM LAN Server?
You Must Have "LAN Intensive Care Utilities"

Without it, what if...
- Alias logon assignments fail?
- Domain/accounts corrupted?
- Permission disappear?

How do you...
- Break-up/consolidate domains?
- Move accounts/resources?
- Administer hundreds of accounts?

See us at OS/2 World in Boston, MA Booth 706

You're dead without our tools!

Lieberman and Associates
221 N. Robertson Blvd. Suite C
Beverly Hills, CA 90211-1703
Phone: 310-550-8575
Fax: 310-550-1152 BBS: 310-550-5980
CS: 76426,363 IBMMAIL: USMVHLVH

CALL OR FAX FOR IMMEDIATE INFO!

Call: 800-829-6263

Try it out yourself today!
Download it from: Our BBS, Compuserve: GO OS2DF2 Lib3, IBMLINK, Hobbes

Ready! for OS/2 Ready! for LAN Server

Circle Reader Service Number 29

You wouldn't use a light-weight development tool to build a business-critical system.
So why would you use a light-weight testing tool to test it?

The Automated Test Facility was built to test the kinds of client/server systems that go directly to a company's bottom line: customer service, claims adjustment, loan processing, inventory management. Systems where reliability and quality are essential.

ATF's unique ability to do multi-user workflow testing has made ATF the testing tool of choice for corporations building business-critical systems under OS/2, Windows, and NT.

Want to hear more? Call 617-576-2257 (or fax 617-864-7747).

ATF
The Softbridge
Automated Test Facility

Softbridge, Inc. ▲125 CambridgePark Drive ▲ Cambridge MA 02140

Circle Reader Service Number 30

The staff of OS/2 Developer *put together this special section to guide you through the various client/server tools. The products in this guide are based on surveys sent to vendors; they are provided as a service to you and are free to vendors. These listings do not represent an endorsement by* OS/2 Developer. *Although every effort has been made to ensure accuracy, we do not assume responsibility for any errors or omissions.*

Client/Server Tools Buyers Guide

ANDERSEN CONSULTING'S FOUNDATION SOFTWARE ORGANIZATION ***Circle No. 101***

Foundation for Cooperative Processing (FCP) allows for asynchronous message routing between client and server systems. FCP is a second-generation client/server solution that allows organizations to develop and deploy enterprise-wide, mission-critical applications. FCP features the Rapid Application Builder (RAB), a design approach that eases the process for developing client/server systems; Foundation Construction, which generates code for client/server applications from objects stored in the repository; and Foundation Production, which includes presentation services, distribution services, and applications services. The current release, FCP 2.4, provides additional capabilities, such as increased ease-of-use through improved online documentation, tutorials, and migration utilities; increased openness through user-defined object types, and a published information model. New productivity features, such as window control and WYSIWYG enhancements; extended architecture for ultrahigh performance (NT clients and an HP-UX gateway to CICS); and new support for OS/2 32-bit, SUN Solaris for SPARC (server), DEC OSF/1 for Alpha AXP (server), and IBM AIX for RS/6000 (server) are also included

Andersen Consulting's Foundation Software Organization, 69 W. Washington, Chicago, Ill. 60602 (312) 507-3322, fax (312) 507-0594.

ASPECT COMPUTING PTY LTD. ***Circle No.102***

LANSA/CS400 develops multilingual client/server applications using a single, easy-to-learn skill set. Its comprehensive OS/2 development environment includes compile, test, debug, and task-tracking facilities. A centralized LANSA repository holds business rules, triggers, and stored procedures and ensures data integrity is maintained. Finished applications have access to data resident on DB2/400, DB2/2 databases, or via ODBC and DRDA interfaces.

Aspect Computing Pty Ltd., Level 11, 122 Arthur St., North Sydney, NSW, Australia, 2060, 61-2-9957-3188, fax 61-2-9957-2657.

BRIGHTWARE ***Circle No. 103***

ART*Enterprise is a client/server application development tool that offers rules, case-based retrieval, and advanced object-oriented programming capabilities for building intelligent applications.

Brightware, 101 Rowland Wy., Ste. 310, Novato, Calif. 94945, (415) 899-9070, fax (415) 899-9080.

CGI SYSTEMS INC. ***Circle No. 104***

PACBASE/CS, is an integrated application development tool that includes support for client/server development and legacy systems. With an interactive multiuser repository driving the process, PACBASE/CS can generate complete applications for over 45 target environments, including IBM mainframes (MVS/VSE), midrange (S/38, AS/400, RS6000), and PCs (OS/2, MS-DOS, UNIX). IBM has selected CGI's PACBASE/CS family of products as its premier I-CASE solution. CGI's integrated application development products operate on mainframes, LANs, and UNIX machines.

CGI Systems, Inc, 1 Blue Hill Plaza, Pearl River, N.Y. 10965, (914) 735-5030, fax (914) 735-2231.

CIMTECH ***Circle No.105***

CIMVIEW is designed for technical management and supervision applications. Based on a distributed real-time and archives database,

it gathers information from all processing equipment (PLCs, DCS, measurement, and so on). In this way, CIMVIEW allows installation managers to completely manage their entire application, either for the daily follow-up and control, or for long-term management such as modeling and quality control.

Cimtech, 20 Rue de L'Industrie, Nivelles, Belgium 1400, 32-067-883666, fax 32-067-883688.

CLIENT/SERVER LABS ***Circle No. 106***

Reference Performance Mark Benchmark (RPMark) is based on industry standard workloads. The RPMark benchmark is a comprehensive performance evaluation designed specifically for client/server environments. It is made up of three workloads running concurrently. The components include: Online Transaction Processing (OLTP), which uses client/server transaction-oriented applications, is based on a subset of the TPC-IP benchmark, and is modified to run in a true client/server environment; Decision Support, which PC clients use to initiate query-intensive applications on the server using Open Database Connectivity (ODBC) calls; and File-Serving/Office Automation Services, based on a subset of the Business Application Performance Corp. (BAPCo) benchmark, for personal productivity applications. The RPMark designation will be given to client/server components undergoing the performance evaluation.

Client/Server Labs, 8601 Dunwoody Pl., Ste. 332, Atlanta, Ga. 30350-2509, (770) 552-3645, fax (770) 993-4667.

COMPUWARE CORP. ***Circle No . 107***

Uniface 6.0 uses a model drama approach to enable developers to build enterprise client/server applications scalability and portability. Uniface 6.0 is based on four principles: model-driven development for high productivity and portability, graphical object-based environment for visual and easy-to-use applications, technology-independent deployment for scalability and flexibility, and team development for group productivity. Uniface 6.0 is a native OS/2 development system that supports all platforms, operating systems, GUIs and character mode, data sources, networks, and CASE tools. Uniface 6.0 supports DB2/2, DB2/6000, DDCS, IBM Lan services, and Workplace Shell.

Compuware Corp., 31440 Northwestern Hwy, Michigan, Ill. 48334-2546, (810) 737-7119, fax (810) 737-7108.

ERGIE S.A. ***Circle No. 108***

TP Tools/2 is an engine that uses services modules for client/server, teleprocessing, and general purposes such as PM screen dynamic and static management, printing models, data access, and communications. Its method is based on rules interpretation, which can be handled by customers, regardless of their development knowledge.

ERGIE S.A., 1 rue Nicolas Copernic, Arles, 13200.

GATEWAY SYSTEMS CORP. ***Circle No.109***

Synergist is a cooperative processing client/server development tool that integrates PCs with host computers. It is used to develop online, transactional-based applications. All applications execute on the PC, accessing a host and/or PC database. Major modules include Dictionary, Forms Generator, and Forms Compiler. Features include pop-up windows, lists, tables, conditional displays, and redefinable function keys.

Gateway Systems Corp., 4660 S. Hagadorn, Ste. 110, East Lansing, Mich. 48823, (517) 337-8960, fax(517) 337-2868.

GLOBAL VILLAGE COMMUNICATION ***Circle No. 110***

FaxWorks API Toolkit provides client/server fax services to applications on OS/2. Applications that you write using API will work with all fax hardware that is supported by the FaxWorks OS/2 products. The API supports both spooled and current call sending and receiving, status monitoring, format conversion, custom dialing, and LAN routing and notification. This product includes the Fax Printer Driver Toolkit.

Global Village Communication, 1110 Northchase Pkwy, Ste. 150, Marietta, Ga. 30067, (404) 984-8088, fax (404)984-9956.

GREAT LAKES SOFTWARE ***CIrcle No. 111***

GLS-Presto is an integrated development environment that lets developers visually create client/server and stand-alone applications, which are executable under both OS/2 and Windows. Presto supports application development in Cobol, C, and C++.

Great Lakes Software, 810 East Coliseum Blvd., Ste. 109, Fort Wayne, Ind. 46805, (219) 481-5809, fax (219) 455-3275.

HOCKWARE ***Circle No. 112***

VisPro/Reports 1.0 is a REXX-enabled report writer for VisPro/REXX, VX REXX, and OS/2 REXX. It allows you to print reports including invoices, form letters, personnel reports, and financial reports. A variety of Avery™ labels are supported, allowing the user to print addresses and other labels. A CUA'91 style WYSIWYG report designer allows the user to include business graphics, photographs, calculated fields, multiline text, and other cosmetic shapes. Apply numerous object attributes, including 16 million colors, OS/2 ATM fonts, fill patterns, and drop shadows.

HockWare, 315 North Academy St., Ste. 100, Cary, N.C. 27513, (919) 380-0616, fax (919) 380-0757.

IBM CORP. ***Circle No. 113***

Paperless Manufacturing Workplace, an OS/2 client/server application, lets you create, maintain, distribute, and display work instructions and associated work orders. Multimedia work instructions are organized so a user views pages of information, displaying the items needed to do the job at hand using media best suited for the task.

PlantWorks is a distributed client/server application development and processing tool for supervisory control and data acquisition (SCADA). It uses the OS/2 Presentation Manager interface and conforms to SAA standards. PlantWorks products use the facilities and functions of the Distributed Automation Editions systems enabler to provide access to the communications and device resources.

IBM Corp., 1000 51st St., Boca Raton, Fla. 33431, (407) 443-9019, fax (407) 443-6824.

IBM CORP. ***Circle No. 114***

VisualGen Developer provides the definition and test of complete applications for client/server, including OS/2 and Windows GUI clients and local and remote servers. VisualGen is a visual programming solution for developing client/server applications. The product includes visual construction of GUI client applications and a robust 4GL for building remote and server applications. Clients are supported on OS/2 and Windows. VisualGen exploits the DB2 and CICS families for data integration and high-volume transaction processing.

IBM Corp., Software Solutions Division, 11000 Regency Pkwy., Bldg. 671/3ba, Cary, N.C. 27511, (919) 469-6329, fax (919) 469-4410.

INTERSOLV INC. ***Circle No. 115***

APS/PC for OS/2 enables rapid development of client/server and traditional production applications for OS/2 with Oracle, MicroSoft/Sybase SQL Server, DB2/2, DB2, DB2/6000, SQL/400, IMS/DM, VSAM, and OS/400 native data file sources. Client/server applications are generated using database servers, gateways, or APPC-connected server programs—100% generated by APS. APS provides an open, extensible, and customizable development environment with no proprietary language or runtime components. APS offers a solution for both new development and existing-system enhancement, for both client/server and traditional systems, across many production platforms.

Intersolv Inc., 1700 N.W. 167th Pl., Beaverton, Ore. 97006, (800) 547-4000, fax (301) 838-5064.

ISA INFORMATIONS-SYSTEME GMBH ***Circle No. 116***

ISA Dialog Manager User Interface Management System supports all major window systems and alphanumeric terminals. A client/server component allows the distribution of application and dialog in heterogeneous environments.

ISA Informationssysteme GmbH, Azenbergstrasse, 35 Stuttgart, D-70174, 49-711-227690, fax 49-711-22769-19.

INTELLIGENT ENVIRONMENTS INC ***Circle No. 117***

AM/Harvest is a project manager and documentation tool that reduces the cost of large-scale client/server application development projects. It offers teams of programmers the capabilities for change-management and "where-used" impact analysis—streamlining application design, construction, and modification. AM/Harvest reads and analyzes program listings, stores the information in a repository, and automatically documents the application. It enables previously developed code to be "harvested" for better maintenance and for reuse in new applications. AM/Harvest generates complete online system documentation in easy-to-use hypertext format (OS/2.INF files). All references to variables, procedures, functions, and modules are hot-linked to details on the usage of each entity.

AM/Solo uses a highly visual, interactive interface to let developers produce client/server applications. AM/Solo is designed to be used by conventional as well as specialist programmers. Its aim is to hide the complexity of graphical user interfaces, multitasking, and object orientation but deliver their power.

Intelligent Environments Inc., 2 Highwood Dr., Tewksbury, Mass. 01876, (617) 272-9700.

KALEIDA LABS INC. ***Circle No. 118***

ScriptX Language Kit (SLK) 1.0 provides technically sophisticated and experienced multimedia content and tool developers early access to the power and flexibility of the ScriptX Language and Class Library. The language is fully object-oriented, multithreaded, device independent, C-extensible, and incrementally compiled. The product also features multiple inheritance, dynamic binding, and built-in data searching. Class Library features include: clocks and players, two-dimensional graphics and text; model-presenter-controller, data management, and user interface or interaction.

Kaleida Labs Inc., 1055-B Joaquin Rd., Mountain View, Calif. 94043, (415) 335-2098, fax (415) 335-2097.

KASE SYSTEMS INC. ***Circle No. 119***

KASE: VIP for OS/2 is a knowledge-assisted visual design and code-generation tool for building GUI and client/server applications. Use the visual designers to create OS/2 objects and then associate them with business logic and data. In a client/server environment, this process produces a seamless integration with SQL databases or CICS transactions. KASE: VIP then generates the application source code in C.

KASE:VIP is a visual design and standard language code-generation tool for developing business-critical client/server applications for OS/2.

KASE Systems Inc., 1 Meca Wy., Ste. 150, Norcross, Ga. 30093, (404) 564-5696, fax (404) 564-5679.

LANWORKS TECHNOLOGIES INC. ***Circle No. 120***

BootWare Manager is a fully integrated Windows-based NetWare LAN management console. It works with BootWare or BootWarePLUS to provide centralized boot management solutions. Administrators can create, maintain, and update boot image files on the server for PCs. BootWare Manager also provides image and global diagnostics with flexible sorting and reporting functions.

Lanworks Technologies Inc., 2425 Skymark Ave., Mississaugia, Ont. L4W 4Y6, Canada (905) 238-5528, fax (905) 238-9407.

MOZART SYSTEMS CORP. ***Circle No. 121***

Mozart, a graphical user interface and software development tool, helps businesses realize immediate client/server benefits from legacy applications. Applications may be developed and run without recompilation or modification. Use Mozart to renovate legacy applications to achieve client/server benefits quickly.

Mozart Systems Corp., 1350 Bayshore Hwy, Ste. 630 , Burlingame, Calif. 94010, (415) 340-1588, fax (415) 340-1648.

NETRON INC. ***Circle No.122***

Netron Fusion is a systems delivery solution for building large-scale applications for mainframe, midrange, and client/server environments. Combined with an integrated toolset, Netron Fusion uses a component approach to software reuse that has been proven to produce core systems 70% faster than the industry average. It features an open, scalable architecture for maximum flexibility and complete control over the development process. It also offers template programs for prototyping, consisting of program specifications, class libraries, screens, and sample files that developers can use immediately.

Netron Inc., 99 St. Regis Cres. N., Toronto, Ont. M3J 1Y9, Canada (416) 636-8333, fax (416) 636-4847.

OPEN SOFTWARE ASSOCIATES INC. ***Circle No. 123***

OpenUI provides a way to develop user interfaces for client/server and standalone applications. It drives the same interface on all character terminals using a standard GUI, without additional development. A user interface developed on one GUI/platform can be migrated without redevelopment to any other GUI/platform, maintaining the native look and feel.

Open Software Associates Inc., 20 Trafalgar Sq., Ste. 414, Nashua, N.H. 03063, (603) 866-4330, fax (603) 598-6877.

PARCPLACE SYSTEMS INC. ***Circle No. 124***

Visual Works is a client/server tool for building portable applications using object-oriented technology. A Database Applications Creator is included for rapid application development. Applications developed are instantly portable across multiple platforms, are scalable across the enterprise, and can have their functionality distributed between both client and servers.

ParcPlace Systems Inc., 999 East Arques Ave., Sunnyvale, Calif. 94086-4593, (408) 481-9090, fax(408) 481-0214.

PROTOSOFT ***Circle No. 125***

Paradigm Plus 3.0 is an application development tool that supports Enterprise Component Modeling (ECM), code generation, and reverse engineering. Using ECM, companies can identify business requirements, model reusable application components, and manage systems over the long term. Paradigm Plus makes ECM possible through its support for leading object-oriented methodologies; incorporation of an object repository for teams of concurrent users; and automatic synchronization of models, source code, and documentation. Using methods such as OMT, Fusion, Martin/Odell OOIE, and others extended to support Jacobson's Use Case modeling, Paradigm Plus automates all leading object-oriented methods in a single solution. Automatic generation of C, C++, Smalltalk, Ada, and ODBMS and RDBMS schema definitions increases productivity. Built-in consistency checks ensure quality and reduce maintenance costs, while a script language allows custom reporting, checking, and code generation. Reverse engineering utilities preserve your investment in legacy software. Online hypertext help, thorough documentation, and hands-on tool training give your team a quick start. Paradigm Plus is available on most popular UNIX and PC platforms.

ProtoSoft, 17629 El Camino Real, Mail Stop 400, Houston, Tex. 77058, (713) 480-3233, fax (713) 480-6606.

SYBASE INC. ***Circle No. 126***

Open Server™ 10.0 is a programmable, network-independent server for creating high-performance client/server applications. It allows client applications to access diverse data, including flat files, real-time data, e-mail, and relational data.

Sybase Inc., 6475 Christie Ave., Emeryville, Calif. 94608, (510) 922-8547, fax (510) 922-4747.

SYNON INC. ***Circle No .127***

Synon Client/Server Generator for OS/2 enables application developers to regenerate Synon/2E design models to run on OS/2 and Windows 3.1 platforms. Synon/CSG delivers distributed function C/S applications with client processing in Microfocus COBOL/2, OS/2 help native communication handling while the AS/400 server stores DB2/400 database.

Synon Inc., 1100 Larkspur Landing Crle., Larkspur, Calif. 94939, (415) 461-5000, fax (415) 461-2171.

TECHBRIDGE TECHNOLOGY ***Circle No. 128***

TechBridge Builder for OS/2, a visual development tool, simplifies creating object-oriented client/server applications through reuse of legacy resources. Developers can paint any GUI, including drag-and-drop, in a visual manner with no coding. TechBridge Builder also offers a COBOL-based scripting language for procedural code attached to GUI events. Developers need not know Smalltalk, C++, or SQL.

TechBridge Visual Objects is a client/server application development tool that simplifies the creation of object-oriented applications. Powerful visual designers can paint any type of GUI, including a drag-and-drop OOUI, with point-and-click connection to SQL and xbase databases. Little or no coding is required. Also supported are: a C-based scripting language with a source debugger and incremental compiler for iterative Rapid Application Development; team support via PVCS interface.

TechBridge Technology, 5001 Yonge St., Ste. 1301, North York, Ont. M2N 6P6, Canada (416) 222-8998, fax (416) 222-0168.

VMARK SOFTWARE INC. ***Circle No. 129***

ENFIN Object Studio is a family of object-oriented products for developing client/server applications based on business objects. These applications are scalable, maintainable, and adaptable to changing processes. ENFIN provides a suite of visual tools for developing and deploying Windows and OS/2 applications. It supports a variety of architectures with a range of connectivity options.

ESL Continuity is an integrated, event-driven, 4GL-based visual tool for developing robust production client/server applications for Windows and OS/2. It translates external source format code from any CASE vendor supporting ESF, including IBM, Texas Instruments, KnowledgeWare, and Bachman, into ESL language source code.

ESL Workbench is an integrated development environment for rapidly creating robust production client/server applications. The application's visual tools for design, debugging, and testing are all grounded in the event-

driven ESL development language. Also included is ESL DB/Assist, a graphical point-and-click tool for generating the SQL portion of your client/server application.

Synchronicity Object Studio is a family of object-oriented products for developing client/server applications based on business objects. These applications are scalable, maintainable, and rapidly adaptable to changing business processes. Synchronicity integrates the design, assembly, and reuse of high-level business objects, ensuring that the company's business, the system design, and the application always remain synchronized. Synchronicity is bidirectionally integrated with ENFIN.

TeamBuilder Object Studio is a family of object-oriented products for developing client/server applications based on business objects. These applications are scalable, maintainable and rapidly adaptable to changing business processes. TeamBuilder enables a development team to jointly build object-oriented client/server applications. Developers can reuse and build upon each other's objects.

VMARK Software Inc., 50 Washington St., Westboro, Mass. 01581, (508) 366-3888, fax (508) 366-3669.

WATCOM ***Circle No. 130***

VX*REXX Client/Server Edition is a visual development environment for creating OS/2 GUI applications. It includes powerful connection, query, and chart objects that access, build, and connect databases, manipulate data, and chart the results quickly and easily.

Watcom, 415 Phillip St., Waterloo, Ont. N2L 3X2, Canada, (519) 886-3700, fax (519) 747-4971.

XVT SOFTWARE INC. ***Circle No. 131***

XVT Development Solution for C++ (DSC++) is a true application framework with powerful features such as field validation, drag and drop, environment inheritance, nested views, geometry management, imaging, and the completely integrated rogue wave data structures. Its visual application builder is used for the entire application development life cycle, from design to layout through final application building. DSC++ offers complete portability to all popular GUIs.

XVT Software Inc., 4900 Pearl East Crle., Boulder, Colo. 80301, (303) 545-3150, fax (303) 443-0969.

Y.A.M. COMPUTERS (1982) LTD. ***Circle No. 132***

OpenWin is an open client/server, 4GL GUI, multiple database, and multiplatform front-end development tool, which applies a distributed applications client/server paradigm. OpenWin is targeted to help developers design front-end user interface components such as screens, windows, flow-control, and field edits. It interfaces GUIs such as OS/2, Windows, and others, while adapting the best of each. OpenWin enables portability of applications across platforms by using local and global dictionaries on LAN servers of differing platforms to simultaneously interface a wide range of DBMS and RDBMS products. OpenWin has an any-to-any client/server ability, with memory and resource sharing, enabling real cooperative processing. Its applications can be used as clients, servers, or both. Using a client/server paradigm to access database engines and user-defined servers, OpenWin is an easy-to-use tool that facilitates modular growth and reduced costs.

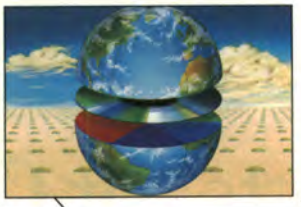
Y.A.M. Computers (1982) Ltd., 2 Hadar St., Herzeliya 46290, Israel, 972-9-589714, fax 972-9-507317.

JBA Guidelines.

The Application Generator for people who use C++ AND those who don't want to.

Database Connectivity

Access to simultaneous, multiple databases is transparent through JOT. Supported databases include: DB2, DB2/2, DB2/400, DB2/6000, Oracle, Lotus Notes Server, and 20 more through native ODBC.

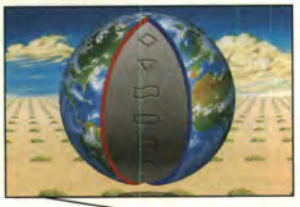


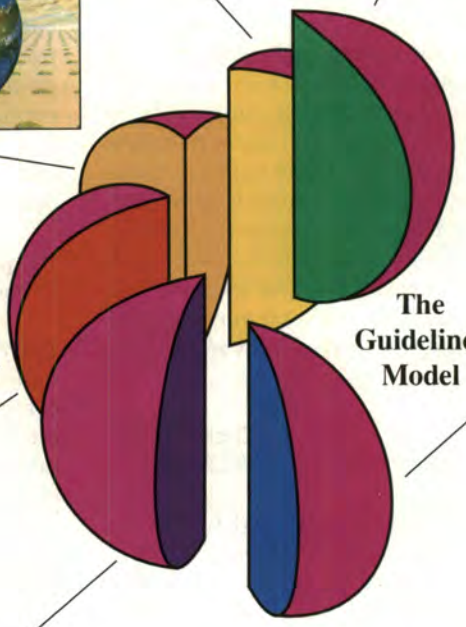
Graphical User Interface

An intuitive and natural interface for the direct manipulation of on-screen design for OS/2 and Windows, with over 30 powerful controls for the professional developer. Includes all the standard controls plus Gauge, Button Bar, Grid and Graph.

Logic, Language, and Compiler

A platform-independent high-level language, JOT combines the RAD features of a 4GL with the generation of native code (C++ and RPG) for the Client **AND** Server. Existing legacy 3GL code can be directly interfaced. And CORBA-compliant objects can replace logic at any time in the future.





The Guidelines Model

The Repository

The repository supports and surrounds the entire programming environment with the facilities for teamwork development. Databases can be created on multiple hosts from a single definition. Data definitions can be directly imported into a GUI program.

Network Connectivity

By mixing the connectivity components as required, applications can be built as thin clients, thick client/server with application partitioning, and fully distributed Object-Oriented. Supported hosts are: OS/2, AS/400, UNIX, MVS.

Object Messaging

The Message layer is fully compatible with all current Object Request Broker (ORB) standards. JOT verbs provide a single interface to current technology and the OO of tomorrow. Guidelines provides CORBA IDL and IBM SOM support.

Constructed on a six-segment development model, JBA Guidelines has been built to answer the needs of developers who wish to produce software capable of addressing heterogeneous computer networks from one single development point.

Guidelines is available in option packs to serve the needs of single PC developers right up to enterprise-wide development teams.

Call for an evaluation CD-ROM.

Available from Indelible Blue
1-800-776-8284

JBA International
1-800-JBA-INTL
In Europe: (44) 1789-400212
Internet: http://www.jba.co.uk
Compuserve: Go Guidelines

Available and Ready™ for OS/2

All trademarks are the property of their respective owners.

©JBA

JBA-161

Circle Reader Service Number 33

Product Watch

The staff of OS/2 Developer *has put together this special section to guide you through the new products. These listings do not represent an endorsement by* OS/2 Developer. *Although every effort has been made to ensure accuracy, we do not assume any responsibility for error in this section. Contact vendor for availability.*

New Products for OS/2

CThrough++. Developed to support the building applications in the C/C++ environment. Instead of working with files, CThrough++ can browse and edit classes and their components–data and functions members. The product stores all information in plain ASCII files and uses an editor to handle them. CThrough++ integrates tools to graphically browse, display, and manage class hierarchies; edits classes in a logical way; creates applications or modules automatically without the need for user-generated files; archives classes and manages different versions; and compares different classes or various versions of one class.

BISS Circle No. 135
Phone: 44-23-92-89-0, Fax: 44-23-92-89-99

Fortran 77 10.5. This enhanced version of Watcom's Fortran 77 development system includes improved productivity with a graphical IDE designed for the development of both character-mode and GUI application on a variety of platforms. This product features the advantages of a 32-bit performance, an advanced GUI debugger, text editor, and profiler. Developers can standardize on a single host environment and toolset for multiplatform development.

Watcom Circle No. 136
Phone: (519) 883-6308

The Graham Utilities 1.04 for OS/2. Version 1.04 now includes a systems diagnostics package for OS/2 systems. This product adds six programs and modules to the already strong suite of over 50 applications. The new utilities add two more HPFS specific programs, two disaster recovery assistance programs, a systems diagnostics package, and a new disk editor module, which allows HPFS disks to be edited in their native format. All of the base HPFS disk structures are able to be edited. The diagnostics program allows testing and reporting on individual aspects of your computer system including, system configuration, parallel and serial port test and configuration, video test including text and graphics test, floppy disk tests, and hard disk and keyboard tests.

WarpSpeed Computers Circle No. 137
Phone: 61-3-9384-1060, Fax: 61-3-9386-9979

KopyKat. KopyKat is a remote-control solution for OS/2. It provides graphical remote control between OS/2 1.3, 2.X, or Warp PCs on LANs with NetBIOS. This product supports hardware handshaking so developers can use high-speed modems at their full capacity. In addition, it can display everything that is displayed on an OS/2 desktop including full-screen DOS and full-screen Win/OS/2 sessions. There are no "don't touch" keys.

Hilgraeve Circle No. 138
Phone: (800) 826-2760, Fax: (313) 243-0645

IPF Builder. This authoring system allows users to create OS/2 INF or HLP files. These files are created by writing "source code" files that are then compiled by IBM's IPFC compiler into either OS/2 viewable INF files or online HLP files. These source code files are called IPF files.

Custom Design Software Circle No. 139
Phone: (604) 334-2263, Fax: (604) 334-3125

IND$file for CICS. This product includes an audit trail for improved security and data compression for better performance. It runs above the 16-MB line, transfers MVS data sets

directly to the workstation, and supports customer-written IND$file exits.

Applied Software Inc. Circle No. 140
Phone: (215) 348-3500, Fax: (215) 348-3535

Interactive System Productivity Facility 4.2 (ISPF). ISPF is a family of licensed programs for the Multiple Virtual Systems (MVS) environment and includes the ability to automatically give existing host applications a GUI without changing, compiling, or distributing any application code or host screens. Version 4.2 offers client/server technology by providing users and programmers with more functions and flexibility for creating high-productivity user interfaces for their applications. In addition, ISPF provides developers with more platforms and communication products from which to access applications. GUI elements, such as group boxes, radio buttons, and drop-down list boxes are included.

IBM Circle No. 141
Phone: (800) 426-2279

Juglar 1.3. This OS/2 test management and defect-tracking tool provides workflow capabilities coupled with reporting facilities to ensure effectiveness for QA managers. Jugalr 13 lets developers produce a structured test plan and produce accurate management information on defect status and activity. In addition, this product reduces the time spent on testing and bug fixing.

Systems FX Circle No. 142
Phone: 44-1844-275175, Fax: 44-1844-343615

LeadTools DLL for OS/2. Offering an imaging solution for compression, file format support, conversion, and image processing, LeadTools DLL for OS/2 contains color, grayscale, and bitonal technology. CMP, Lead's bitmap format, features enhanced image-processing features, smaller file size, and faster decompression. This dynamic link library supports OS/2 2.1 and OS/2 Warp and can be used to enhance applications such as image databases, printing and drawing programs, video production systems, FAX systems, and systems for transmitting high-quality images using phone lines and networks. LeadTools lets the user convert to and from a variety of image file formats including OS/2's native BMP formats, copy and paste with the clipboard, and print.

Lead Technologies Inc. Circle No. 143
Phone: (704) 332-5532, Fax: (704) 372-8161

OS/2 Warp Developer's Package for Pentium. This package includes NDP Fortran, NDP Fortran 90, NDP Pascal, and NDP C/++. In addition, this product works with the IBM Workframe. NDP Fortran is a full Fortran 77 with BSD 4.2 and DOD extensions. It is 99% VAX/VMS compatible. NDP C/C++ is a full AT&T 2.1 compliant compiler that provides optimizations for numeric-intensive application, and NDP Pascal is an ANSI/IEEE compliant Pascal that also passes BSI Level 0 and can access most of the NDP C run-time functions.

MicroWay Circle No. 144
Phone: (508)746-7341, Fax: (508) 746-4678

ObjectCatalog. A distributed, cross-platform component reuse facility, ObjectCatalog enables different development teams to share information about software, design patterns, frameworks, documents, and other corporate assets. ObjectCatalog allows the user to fill out questionnaires and define search patterns for finding target entries in local and remote catalogs. Matching entries are then presented graphically according to their degree of similarity to the request.

ObjectSpace Inc. Circle No. 145
Phone: (214) 934-2496, Fax: (214) 663-3959

ObjectPM Control Pack Library for OS/2. This package offers over a dozen control types that extend the set of controls supplied by OS/2 Presentation Manager. It also supports the PMCX control window specification, allowing these controls to be used with products such as the IBM Universal Resource Editor and Prominare Designer. The PMCX specification is the latest control extension, similar in concept to the VBX specification in Windows. Included in this package are a spreadsheet, cellbox, data field, RTF viewer, calendars, and splitbars. All controls in ObjectPM Control Pack conform to the PMCX architecture, allowing the components to be used by any OS/2 programming tool.
Secant Technologies Inc. Circle No. 146
Phone: (216) 595-3830, Fax: (216) 292-2546

Paradigm Plus 3.0. This object-oriented development tool supports the entire software development lifecycle. Its graphical environment automates leading object-oriented methodologies and notations, including OMT, Booch, Fusion, Martin/Odell, Caod/Yourdon, and Shlaer/Mellor, while also allowing the developer to customize methods based on business requirements. Paradigm Plus generates code in C, C++, Ada, and Smalltalk, as well as RDBMS and ODBMS schema definitions. Utilizing an advanced object repository and open architecture, the product optimizes reuse by allowing access to all stored objects and relationships in a multiuser environment. Unlike other object-oriented tools currently available on the market, Paradigm Plus facilitates reverse engineering of existing development efforts.

ProtoSoft Inc. Circle No. 147
Phone: (713) 480-3233, Fax: (713) 480-6606

PartitionMagic. This product will let developers visually modify a hard disk on screen as well as shrink, expand, move, and convert partitions without destroying data. PartitionMagic automatically converts a FAT file system to HPFS, creates room for Boot Manager either before or after OS/2 is installed, and combines unused portions of several partitions into a larger block.

PowerQuest Corp. Circle No. 148
Phone: (800) 379-2566, Fax: (801) 226-8941

RhinoCom 1.5. RhinoCom's Workplace Shell interface replaces the dialing directory with an intuitively object-oriented phone book, or Rhinodex. This product includes extensive configurability with logical defaults, full-featured macros, fast transfers, and REXX scripting. Enhanced features include, automated script learning with AutoLogon, graphical manipulations of global settings, host mode as a secure background BBS, and the CompuServe B+ transfer protocol.

Rhintek Inc. Circle No. 149
Phone: (410) 730-2575, Fax: (410) 730-5960

Velocis 1.3.1. This product offers software developers access to tools for creating high-performance client/server applications. New features include transaction-processing performance improvements, robust dynamic control over low-disk space conditions, improved server configuration capabilities, and additional administration API functions. Use of the server extensions allows three-tier architecture, which can reduce network traffic and take advantage of server power by partitioning application logic between client and server. This product also supports server extensions created in C++.

Raima Corp. Circle No. 150
Phone: (206) 557-0200, Fax: (206) 557-5200

Show N Tel 3.0. This object-oriented development environment for building and implementing telephony systems enables interactive voice, fax, call processing, speech recognition, call center, and multimedia messaging applications. This release adds a range of new functions for corporate developers, system integrators, and OEMs, including over 100 PowerBlocks–graphical program design objects representing high-level application functions. Show N Tel includes a palette of over 300 PowerBlocks, with new objects for tighter PBX and switch control, in-chassis switching, and text-to-speech, messaging, and interactive fax functions.

Technically Speaking Circle No. 151
Phone: (508) 229-7777, Fax: (508) 229-8777

Spoolview. Spoolview, a product released by Datatrade, now supports OS/2 client and server platforms. It is a multiplatform application that collects reports from all IBM host systems via file transfer or tape input. The reports are then automatically indexed, compressed, and transferred to IBM 3995 optical media. Users have the ability to view, print, fax, and export data via LAN-based PCs running OS/2. Native IBM AFP advanced print streams are fully supported.

Datatrade Circle No. 152
Phone: (417) 882-1576, Fax: (417) 882-8423

Snow Report Writer. Part of Snow's cross-platform family of client/server report-writing solutions, this product uses the common word processor metaphor to design, format, and place data in reports. The product incorporates second-generation report-writing features. The Snow engine provides report-writing capabilities without requiring the knowledge of an underlying data access language, such as SQL.

Snow Software Circle No. 153
Phone: (813) 784-8899, Fax: (813) 787-1904

AHA-2940 Ultra Wide. The AHA-2940 Ultra Wide single-channel PCI-to-SCSI host adapter offers faster I/O throughput and improved efficiency for PCI-based systems by speeding data exchange between SCSI peripherals and the PCI system at speeds of up to 40 MB/second. In addition, the adapter's flexible design allows customers to utilize existing SCSI peripherals, connections, and cabling, protecting their existing investment. This product can also support the mixing of 8- and 16-bit devices on the same SCSI cable, without affecting the reliable operation of the system.

Adaptec Circle No. 154
Phone: (408) 262 8600, Fax: (408) 262-2533

UniMaint 4.0. UniMaint for OS/2 uninstalls any OS/2 application, including freeware, shareware, or commercial software. In addition, UniMaint maintains OS/2 INI files and extended attributes while also performing desktop backups or portable desktop backups to port an OS/2 desktop to your laptop computer or your machine at home.

SofTouch Systems Inc. Circle No. 155
Phone: (405) 947-8080, Fax: (405) 632-6537

Bring enterprise-wide alphanumeric paging to your network and monitor mission critical applications across the enterprise with SkyAgent™. Save time, money and trim your I/S paging support costs by consolidating your enterprise's outbound paging operations into a single paging center. SkyAgent can be used to monitor system processes for predetermined events - which in turn can automatically page the appropriate network support staff.

Through leading edge TCP/IP SNMP technology and SkyAgent's SNMP extensible "paging" agent, every SNMP-ready host on your LAN or WAN has the capability to send multiple text messages to multiple pager devices in multiple locations with a single host SNMP command.

Get important messages from all your computer systems to the right people in your organization at the right time, 24 hours a day, 7 days a week, whether they're in New York, Los Angeles, or on an airplane somewhere in between. With SkyAgent, you avoid having to install paging software on every computer system in your enterprise. Plus, you get the ultimate, full-function paging server for all hosts on your network.

You can ensure your mission critical apps are well taken care of - when you connect your best LAN support people with SkyAgent.

Copyright GammaTech Inc. 1995

- Industry standard SNMP agent interface means sending requests to a SkyAgent server is as easy as issuing a single command from a host out across your network
- Connectability to an unlimited number of "pager company" service providers through multiple, simultaneous, T.A.P. modem connections means you can reach anyone in any provider's service areas
- High volume message queuing and delivery rates can handle 500,000+ messages per month from hundreds of different systems
- Built-in user data base allows messages to be sent to nicknames and groups so that no one has to remember pager numbers -- This data base may be updated remotely via SNMP requests to lessen system administration costs
- SkyAgent standardizes your enterprise's computer-based paging and lowers paging support costs for your organization
- Complete logging, tracing, and diagnostic options ensure system and message accountability
- SkyAgent is designed around IBM's OS/2 Warp operating system and is a full 32-bit, multitasking server process
- SkyAgent is designed to run in an unattended environment and may be monitored remotely from any type of SNMP "network management station"
- MIB definition files are included for most common host platforms
- Optionally purchase all hardware, configured "ready for use", and simply plug it into your network for immediate computer-based paging service

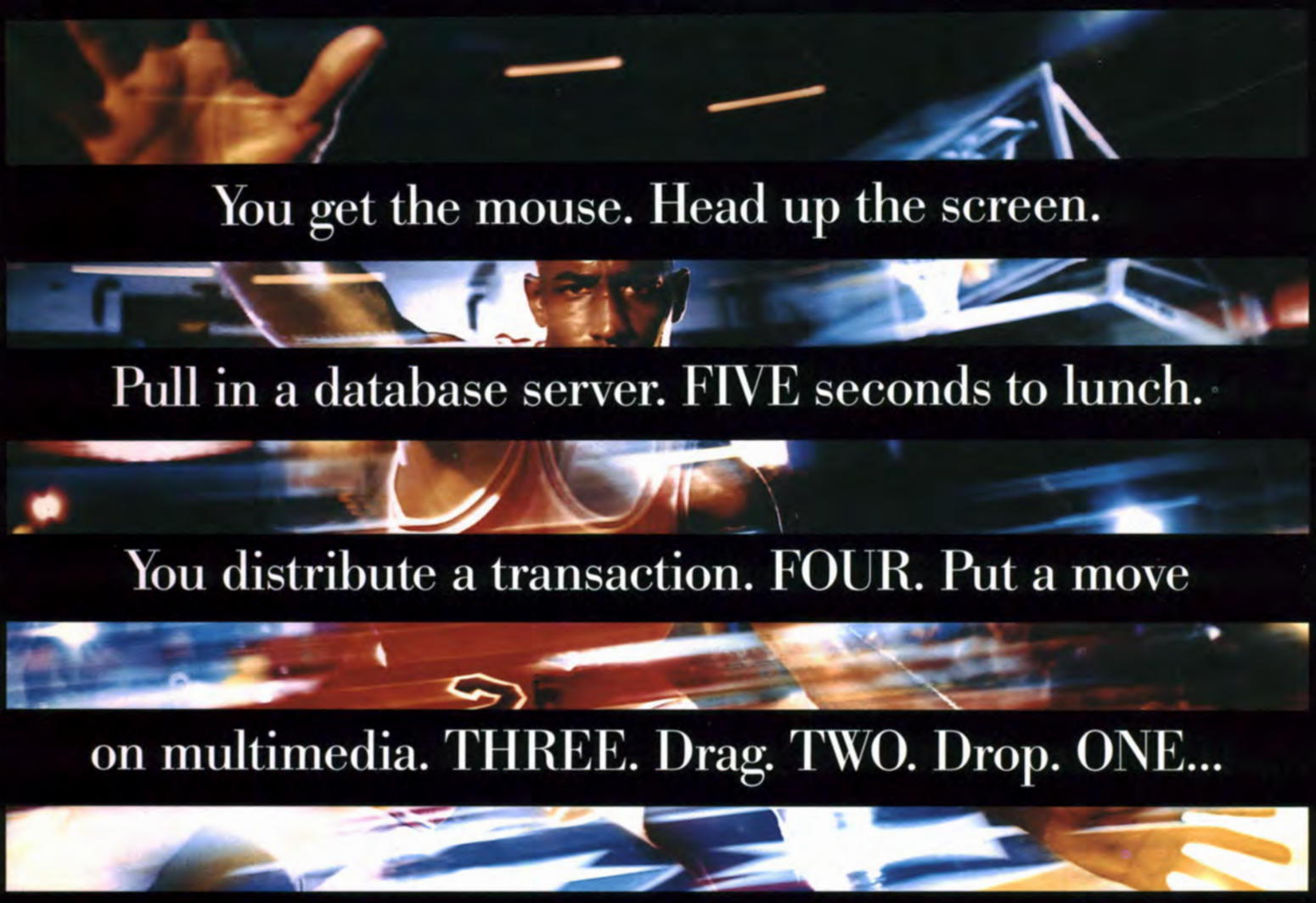
SKYAGENT™

(800) 944-3028 STS **FAX (405) 632-6537**

SofTouch Systems, Inc.

1300 S. Meridian, Suite 600, Oklahoma City, OK 73108-1751

Circle Reader Service Number 32



Yeeesssss. It's all over, baby.

It's easy to see why nothing propels you into the big leagues of object-oriented programming like our new VisualAge™ C++ version 3.0.

It transcends mere GUI builders. ***Can your software do this?*** This fully integrated development environment lets you generate tight, fast client/server applications with point-and-click ease and efficiency.

A simple drag-and-drop incorporates the truly scalable capabilities of a vast library of pre-built Open Class objects, which means you can create distributed client/server apps quickly. And with Open Class and C++ compilers for OS/2®, Sun™ Solaris,™ OS/400®, AIX®, and MVS, deploying your new object-oriented apps across multiple platforms is really easy. We thought that new features like these would excite you. *Infoworld* agreed, and has called our new VisualAge C++ "object reusability at its finest" and "a masterpiece of visual programming."†

True OO client/server development with VisualAge C++ for OS/2. It could be the best move you make.

Get a Developer's Kit including OS/2 Warp and a CD featuring an evaluation copy of DB2® v2 when you buy VisualAge C++ for OS/2.* Call your authorized IBM reseller or 1 800 3IBM-OS2, Dept. SA019.

IBM®
Solutions for a small planet.™

*This is a limited-time offer. To receive your Developer's Kit, simply complete and return the VisualAge C++ registration card by December 29, 1995. OS/2 Warp will be shipped on CD ROM only. In Canada, please call 1 800 IBM-CALL, ext. 8012. Outside North America, please contact your local IBM office. VisualAge C++ for OS/400 will be available in 4th Qtr., 1995. The IBM software home page is located at http://www.software.ibm.com. IBM, OS/2, DB2, OS/400 and AIX are registered trademarks and VisualAge is a trademark of International Business Machines Corporation. All other trademarks are the property of their respective owners. †*Infoworld* review, 5/8/95. © 1995 IBM Corporation. All rights reserved.

Circle Reader Service Number 31